VŠB-Technical University of Ostrava
Faculty of Mechanical Engineering
Department of Applied Mechanics

Solution of Torsion of beams with Non-Circular Cross- Sections in Python

Řešení úloh kroucení prutů nekruhových průřezů v Pythonu

Student: Bc. Tea Tutiashvili
Supervisor: Ing. Alexandros Markopoulos, Ph.D

Ostrava 2016

VŠB - Technical University of Ostrava
Faculty of Mechanical Engineering
Department of Applied Mechanics

Diploma Thesis Assignment

Student: **Bc. Tea Tutiashvili**

Study Programme: N2301 Mechanical Engineering

Study Branch: 3901T003 Applied Mechanics

Title: Solution of Torsion of Beams with Non-Circular Cross-Sections in Python
Řešení úloh kroucení prutů nekruhových průřezů v Pythonu

The thesis language: English

Description:

1. Application of deformation variant of FEM.
2. Utilizing triangular and rectangular elements for the discretization.
3. Development of own FEM software written in Python.
4. Own mesh generator for basic cross-sections (squared, triangular, circular, elliptical)
5. Option to solve general cross-section prepared in other software (ANSYS or Patran)
6. Validation of numerical results via cases for which analytical solution is known.

References:

1. Theory of Elasticity, S.P. Timoshenko, J.N. Goodier, ISBN-13: 978-0070647206
2. Archie Higdon et al. "Mechanics of Materials, 4th edition".
3. Advanced Strength and Applied Elasticity, Ugural & Fenster, Elsevier, ISBN 0-444-00160-3

Extent and terms of a thesis are specified in directions for its elaboration that are opened to the public on the web sites of the faculty.

Supervisor: **Ing. Alexandros Markopoulos, Ph.D.**

Date of issue: 11.12.2015
Date of submission: 16.05.2016

doc. Ing. Radim Halama, Ph.D.
Head of Department

doc. Ing. Ivo Hlavatý, Ph.D.
Dean of Faculty

Declaration of the student

I hereby declare that this master's thesis was written by myself. I have quoted all the references. I have drawn upon.

Ostrava 16.05.2016 Student's signature 

Prohlašuji, že

- jsem byl seznámen s tím, že na moji diplomovou (bakalářskou) práci se plně vztahuje zákon č. 121/2000 Sb., autorský zákon, zejména § 35 – užití díla v rámci občanských a náboženských obřadů, v rámci školních představení a užití díla školního a § 60 – školní dílo.
- beru na vědomí, že Vysoká škola báňská – Technická univerzita Ostrava (dále jen „VŠB-TUO") má právo nevýdělečně ke své vnitřní potřebě diplomovou (bakalářskou) práci užít (§ 35 odst. 3).
- souhlasím s tím, že diplomová (bakalářská) práce bude v elektronické podobě uložena v Ústřední knihovně VŠB-TUO k nahlédnutí a jeden výtisk bude uložen u vedoucího diplomové (bakalářské) práce. Souhlasím s tím, že údaje o kvalifikační práci budou zveřejněny v informačním systému VŠB-TUO.
- bylo sjednáno, že s VŠB-TUO, v případě zájmu z její strany, uzavřu licenční smlouvu s oprávněním užít dílo v rozsahu § 12 odst. 4 autorského zákona.
- bylo sjednáno, že užít své dílo – diplomovou (bakalářskou) práci nebo poskytnout licenci k jejímu využití mohu jen se souhlasem VŠB-TUO, která je oprávněna v takovém případě ode mne požadovat přiměřený příspěvek na úhradu nákladů, které byly VŠB-TUO na vytvoření díla vynaloženy (až do jejich skutečné výše).
- beru na vědomí, že odevzdáním své práce souhlasím se zveřejněním své práce podle zákona č. 111/1998 Sb., o vysokých školách a o změně a doplnění dalších zákonů (zákon o vysokých školách), ve znění pozdějších předpisů, bez ohledu na výsledek její obhajoby.

Ostrava 16.05.2016

.....

Signature

Name and surname of the author's work: Bc. Tea Tutiashvili

Permanent address of author: Samghereti st. 34 Tbilisi 0101 Georgia

Acknowledgement

At this point I would like to acknowledge and express my special thank the supervisor of my thesis Ing. Alexandros Markopoulos, Ph.D. Who has encouraged and supported me throughout this entire process. Also I would like say thank doc. Ing. Radim Halama Ph.D. for help to calculate some special thing in program ANSYS.

Anotace diplomové práce

TUTIASHVILI, T. *Řešení úloh kroucení prutů nekruhových průřezů v Pythonu.* Ostrava: VŠB-Technická univerzita Ostrava, Fakulta strojní, Katedra aplikované mechaniky, 2016, 75 s. Vedoucí práce: Markopoulos, A.

Diplomová práce se zabývá problematikou volného kroucení. Původně 3 dimenzionální úloha je zjednodušena na dvojrozměrný problém, což je možné díky ponechání volné deplanace na průřezech (použití tzv. membránové analogie). Následující diferenciální rovnice rovnováhy je řešena metodou konečných prvků ve vlastní knihovně napsané v jazyce Python. Kvalita řešení je ověřena na průřezech, pro které je známo analytické řešení. V práci jsou odvozeny matice tuhosti a pravé strany pro rovinné konečné prvky (trojúhelníkový a čtyřúhelníkový prvek). Tyto prvky jsou implementovány ve vlastní knihovně psané v jazyce Python, v níž byla spočtena většina problémů obsažených v této práci.

Annotation of master's thesis

TUTIASHVILI, T. *Solution of Torsion of beams with Non-Circular Cross- Sections in Python.* VŠB-Technical University of Ostrava, Faculty of Mechanical Engineering, Department of Applied Mechanics, 2016, 75 p. Head of thesis: Markopoulos, A.

This thesis deals with unconstrained torsion of prismatic beams. Original 3 dimensional problem is simplified to 2 dimensional due to free warping (technique known as membrane analogy). Followed equilibrium equation is solved by Finite Element Method in own library written in Python. The quality of the solution is validated for the cases of cross-sections with known analytical solution. The work includes derivation of stiffness matrix and right-hand side for 3 and 4-nodes planar elements. These elements are implemented in own library written in Python in which most of numerical results were performed.

Contents

List of symbols and abbreviations

T	Torque [Nm].
l	Length [m].
J	Polar moment of inertia (Non circular sections) [m^4].
R	Radius of section [m].
τ	Shear stress [N/m^2].
G	Modulus of elasticity in shear. Modulus of rigidity [N/m^2].
θ	Angle of twist [radians].
x, y, z	Coordinates.
A	Cross-sectional area.
ρ	Density.
M	Bending moment.
$\sigma_x, \sigma_y, \sigma_z$	Normal components of stress parallel to x-, y-, and z-axes.
$\tau_{xy}, \tau_{xz}, \tau_{yz}$	Shear-stress components.
u, v, w	Components of displacement.
$\varepsilon_x, \varepsilon_y, \varepsilon_z$	Unit elongations in x-, y-, and z-directions.
γ	Unit shear.
$\gamma_{xy}, \gamma_{xz}, \gamma_{yz}$	Shear strain components.
E	Modulus of elasticity.
μ	Poisson's ratio.
Φ	Stress function.
X, Y, Z	Components of a body force per unit volume.
$\overline{X}, \overline{Y}, \overline{Z}$	Components of a distributed surface force per unit area.
l, m, n	Direction cosines of the outward normal.
r, θ	Polar coordinates.
e_x, e_y, e_z	Unit elongations in x-, y-, and z-directions.

List of figures

List of tables

1. Introduction

The cases of beam torsion with common cross-section, are frequent mathematical problems in engineering. Finding the analytical solution is often very complicated and sometimes even impossible. For these reasons, the difficult cases are solved by numerical methods, usually by the finite element method (FEM). FEM was applied to the Saint-Venant's theory of free torsion (warping is allowed).

At the beginning, the theory of torsion is introduced to get required partial differential equations. Due to the fact, the torsion is free, the general 3 dimensional case can be reformulated to 2 dimensional one, depending on just two variables (x, y). The numerical results are presented in the own software written in Python. The functionality of the software is verified according to the cross-sections (circular, triangular, square and rectangular) for which analytical solution is known. At the end of this thesis, the realistic 3 dimensional problem with special cross-section is analyzed in ANSYS, and the results are compared the application written in Python.

2. Torsion of beam generally

2.1. Torsion of circular cross-sections

The circular cross-section loaded by torsion moment is a suitable problem for validation of numerical methods. In next part the detailed theoretical background of such problem will be described.

2.1.1. Stress and strain distributions

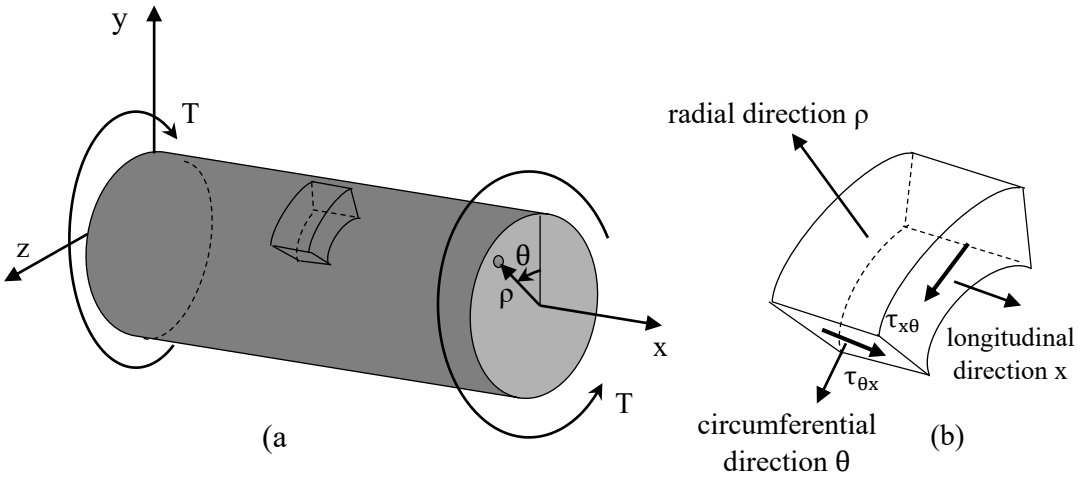


Fig. 1 Extracting a (x, ρ, θ)-aligned material element. The only nonzero stresses are $\boldsymbol{\tau_{\theta x} = \tau_{x\theta}}$. **[1]**

From the shaft shown on Fig. 1 left a material small element (right) is extracted, which is magnified in Fig.1(b), which plainly shows that the only nonzero stress component on its faces is the shear stress $\tau_{x\theta} = \tau_{\theta x}$. It follows that the stress and strain matrices referred to the cylindrical coordinate system have the form

$$\begin{bmatrix} 0 & \tau_{x\theta} & 0 \\ \tau_{\theta x} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \leftarrow \text{Hooke's law} \rightarrow \begin{bmatrix} 0 & \gamma_{x\theta} & 0 \\ \gamma_{\theta x} & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}. \tag{1}$$

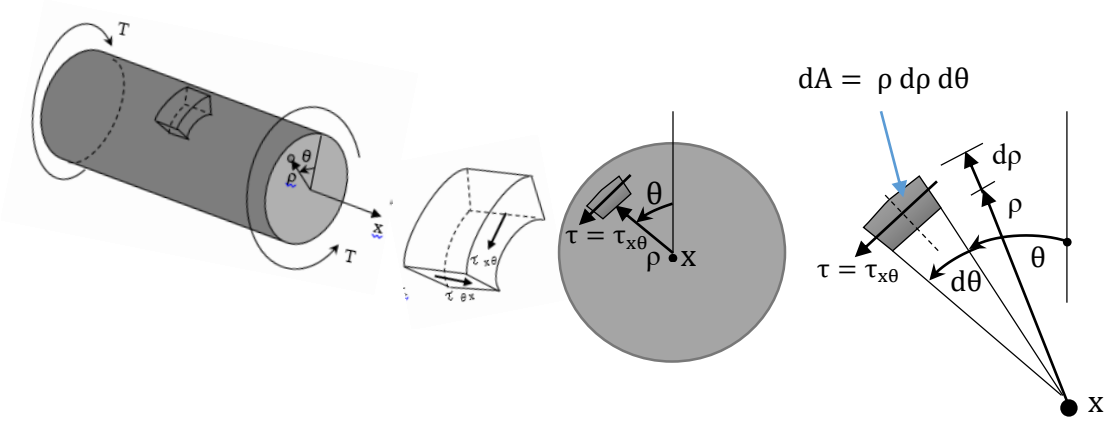


Fig. 2 Integration over the cross-section of the elementary moment produced by the shear stress component $\boldsymbol{\tau = \tau_{x\theta}}$ will balance the internal torque T. **[1]**

The axes have been reordered as (x, θ, ρ) for convenience in identifying with plane stress and strain later. In the sequel we often call $\tau = \tau_{x\theta} = \tau_{\theta x}$, and $\gamma = \gamma_{x\theta} = \gamma_{\theta x}$ to reduce subscript clutter. Stresses and strains in (1) are connected at each material point by Hooke's shear law

$$\tau = G\,\gamma, \quad \gamma = \frac{\tau}{G} \tag{2}$$

in which G is the shear modulus. Because of circular symmetry, τ and γ do not depend on θ. They may depend on *x* if either the cross-section radius or the internal torque change along the shaft length, but that variation will not be generally made explicit. We will assume (subject to a posteriori verification) that τ and γ depend linearly on ρ, and we scale that dependence as follows:

$$\tau = \frac{\rho}{R}\tau_{max}, \quad \gamma = \frac{\rho}{R}\gamma_{max}. \tag{3}$$

Here τ_{max} and γ_{max} are maximum values of the shear stress and strain, respectively, over the cross-section. These occur at ρ = R, the radius of the shaft (for an annular shaft, R = Re, the exterior or outer radius). More details you can find in [1]

2.1.2. The stress formula

Call T the internal torque acting on a cross-section x = constant. The shear stress $\tau = \tau_{x\alpha}$ acting on an elementary cross-section area dA produces an elementary force dF= τ dA and an elementary moment $dM = dF_\rho = (\tau\, dA)\,\rho$ about the x axis. See Fig. 2 for the geometric details. Integration of the elementary moment over the cross-section must balance the internal torque:

$$T = \int_A (\tau dA)\rho = \int_A \frac{\rho}{R}\tau_{max}\rho dA = \frac{\tau_{max}}{R}\int_A \rho^2 dA = \frac{\tau_{max}}{R}\int_A \rho^2 (\rho d\rho d\alpha)$$
$$= \frac{\tau_{max}}{R}\int_A \rho^3 \, d\rho d\alpha = \frac{\tau_{max}}{R} J, \tag{4}$$

in which

$$J = \int_A \rho^2 \, dA = \int_A \rho^3 \, d\rho d\alpha \tag{5}$$

is the polar moment of inertia of the cross-section about its center. Solving for τ_{max} gives the stress formula:

$$\tau_{max} = \frac{TR}{J}, \; \tau = \frac{\rho}{R}\tau_{max} = \frac{T\rho}{J}, \tag{6}$$

for a solid circular section of radius R, $J = \frac{1}{2}\pi R^4$. For an annular cross-section of exterior radius Re and interior radius Ri, $J = \frac{1}{2}\pi(R_e^4 - R_i^4)$. More details you can find in [1]

2.1.3. The twist rate and angle formulas

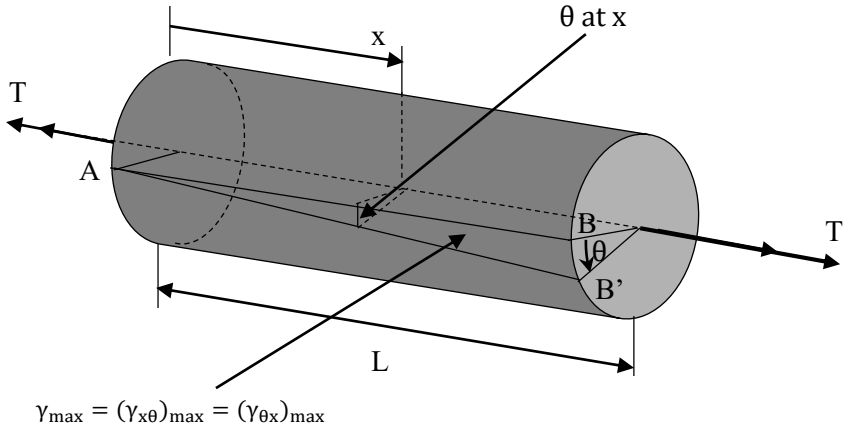


Fig. 3 Twist deformation of a torqued circular shaft. Deformations and rotations greatly exaggerated for visualization convenience. **[1]**

A torqued shaft produces a relative rotation of one end respect to the other; see visualization in Fig. 4.

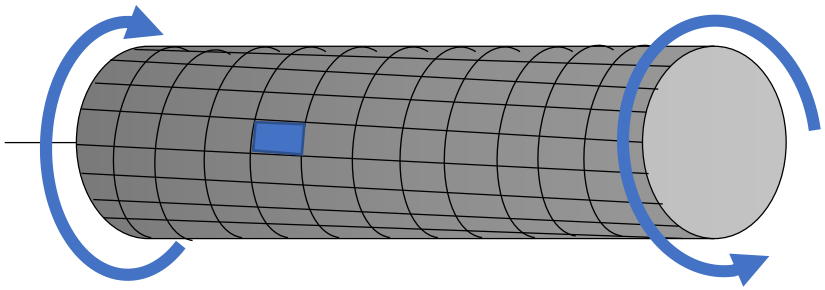


Fig. 4 The deformation of the circular shaft produced by torque moment. **[1]**

This relative rotation is called the twist angle θ. The twist angle per unit of x-length is called the twist rate $\frac{d\theta}{dx}$. Often the design called for limitation on this rate or angle. It is thus of interest to have expressions for both of them in terms of the applied torque and the properties of the shaft. Geometric quantities used for working out those expressions are defined in Fig. 3. Note that infinitesimal strains and rotations are assumed; deformations depicted in Fig. 3are greatly exaggerated for visibility convenience. For small angles $BB' \approx R\theta_{BA} \approx L\gamma_{max}$. At an arbitrary distance x from the fixed end, $R\,\theta \approx x\,\gamma_{max}$ whence $\theta = x\,\gamma_{max}/R$. Consequently the twist rate is

$$\frac{d\theta}{dx} = \frac{\gamma_{max}}{R} = \frac{\gamma}{\rho} \text{ so } \gamma = \rho\frac{d\theta}{dx} \tag{7}$$

but according to Hooke's law (2)

$$\gamma = \frac{\tau}{G} = \frac{T\rho}{GJ} = \rho\frac{d\theta}{dx}, \tag{8}$$

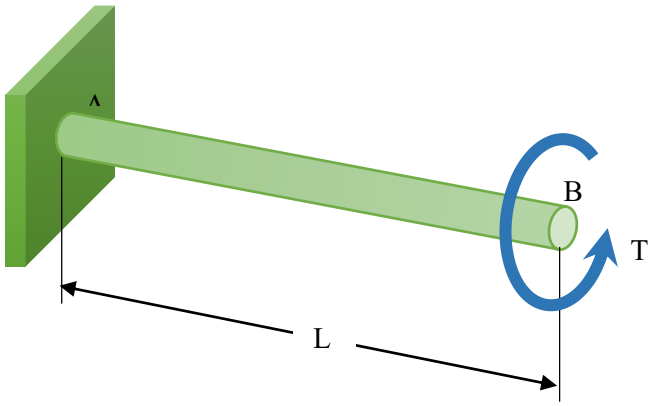


Fig. 5 The scheme of solved problem, location of cross-sections A, and B. **[1]**

which yields

$$\frac{d\theta}{dx} = \frac{T}{GJ}, \tag{9}$$

this is the twist rate formula. To find the twist angle θ_{BA}, where subscripts identify the angle measurement endpoints, integrate along the length of the shaft:

$$\theta_{BA} = \theta_B - \theta_A = \theta_B - 0 = \theta_B = \int_0^L d\theta = \int_0^L \frac{d\theta}{dx} dx = \int_0^L \frac{T}{GJ} dx, \tag{10}$$

if T, G and J are constant along the shaft:

$$\theta_{BA} = \frac{T}{GJ} \int_0^L dx = \frac{TL}{GJ}, \tag{11}$$

this is the twist angle formula for a prismatic shaft. More details you can find in [1]

2.1.4. Principal stresses and maximum shear

We know that the stress matrix at a point P (x, ρ, θ) referred to a cylindrical coordinate system with axes reordered as (x, θ, ρ) takes the simple form

$$\begin{bmatrix} 0 & \tau & 0 \\ \tau & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix}, \tau = \tau_{\theta x} = \tau_{x\theta} = \tau_{max} \frac{\rho}{R}, \tau_{max} = \frac{TR}{J}. \tag{12}$$

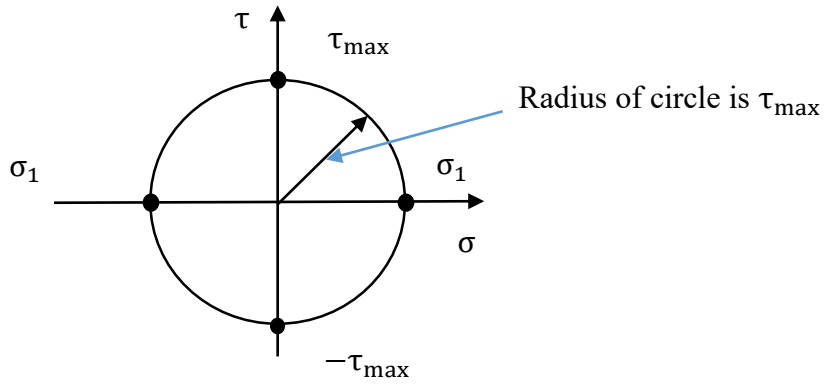
We can observe that all points are in plate stress because all stresses with a ρ subscript vanish, although the shaft is not a thin plate. It is also in plane strain because the transverse strains (that is, strain components with a ρ subscript): $\varepsilon_{\rho\rho}, \gamma_{x\rho}$ and $\gamma_{\theta\rho}$, are zero.

Since the shear stress is proportional to ρ, we consider a point on the outer surface ρ = R, where $\tau = \tau_{max}$. The Mohr's circle for stresses in the (x, θ) plane passing through $P(x, R, \theta)$is shown in Fig. 6(a). By inspection we see that

(a) The principal stresses are $\sigma_1 = \tau_{max}$(tension) and $\sigma_2 = -\tau_{max}$ (compression). These occur at± 45° from the longitudinal x direction.

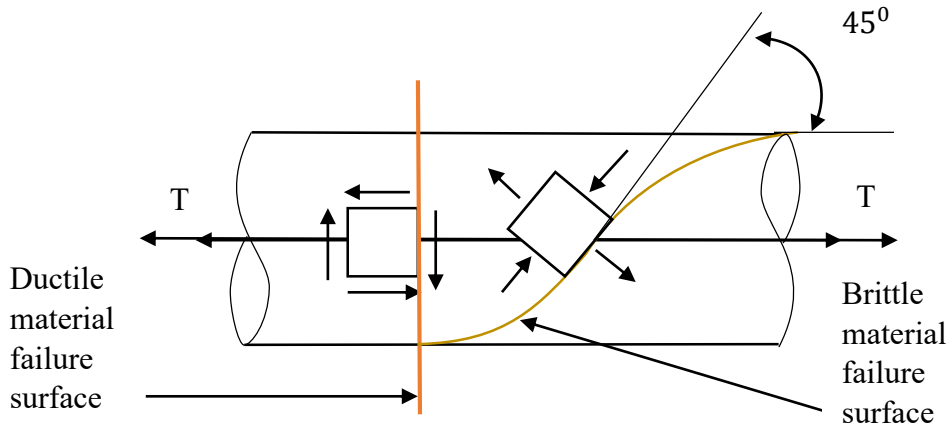
(b) The maximum in plane shear is τ_{max}, which occurs on the shaft cross-sections.

Consideration of 3D effects does not change these findings because the zero principal stress is the intermediate one. Thus the circle plotted in Fig. 6(a) is the outer one, and the maximum overall shear stress $\tau_{max}^{overall}$ is the same as the maximum inplane shear stress, that is, $\tau_{max}^{inplane}$. More details you can find in [1].



(a)

(a). Mohr's circle for (x, **θ**) plane at **ρ = R** (outer shaft surface). **[1]**



(b). Material failure surfaces for two extreme cases: ductile vs. brittle. [1]

Fig. 6 By inspection of the Mohr circle, maximum normal and shear stresses at a point on the outer shaft surface determine the kind of material failure. **[1]**

2.2. Torsion of a non-circular prismatic bars

Take, for instance, a bar of rectangular cross-section (Fig. 7). From Navier's assumption it follows that at any point A on the boundary the shearing stress should act in the direction perpendicular to the radius OA.

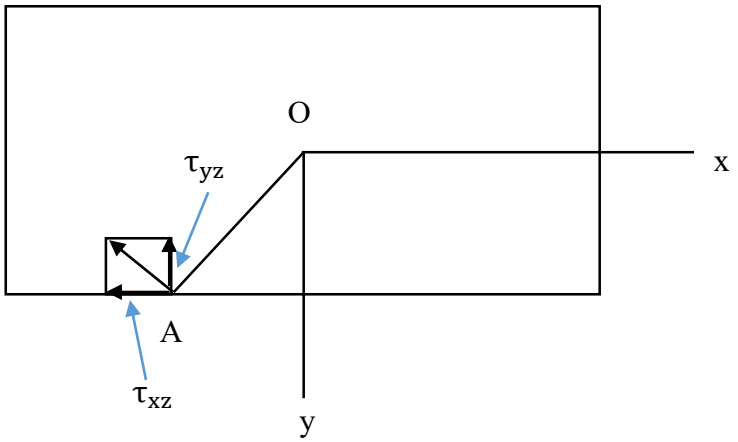


Fig. 7 A bar of rectangular cross-section. **[2]**

Resolving this stress into two components τ_{xz} and τ_{yz}, it is evident that there should be a complementary shearing stress, equal to τ_{yz}, on the element of the lateral surface of the bar at the point A, see Fig. 8,

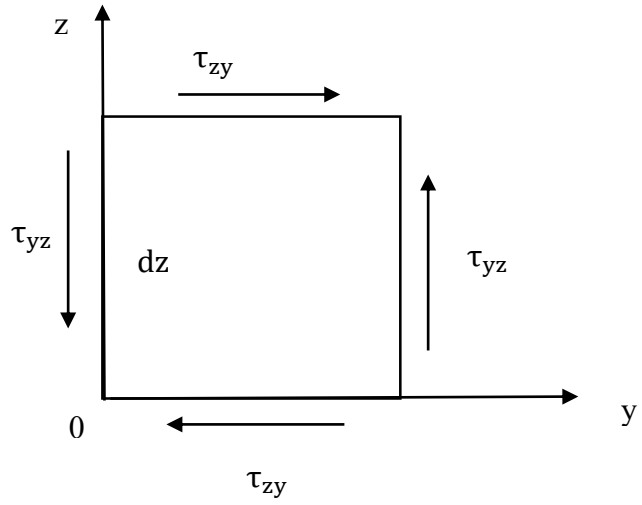


Fig. 8 The moments of the forces acting on the element about the x-axis. **[2]**

which is in contradiction with the assumption that the lateral surface of the bar is free from external forces, the twist being produced by couples applied at the ends.

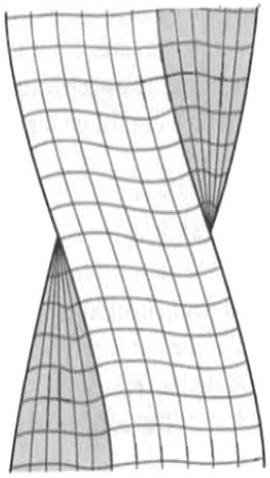


Fig. 9 The distortions of rectangular elements on the surface of the bar are greatest at the middles of the sides. **[2]**

A simple experiment with a rectangular bar, represented in Fig. 9, shows that the distortions of rectangular elements on the surface of the bar are greatest at the middles of the sides, i.e., at the points which are nearest to the axis of the bar.
The correct solution of the problem of torsion of prismatical bars by couples applied at the ends was given by Saint-Venant.

He used the so-called semi-inverse method. That is, at the start he made certain assumptions as to the deformation of the twisted bar and showed that with these assumptions he could satisfy the equations of equilibrium

$$\begin{aligned} \frac{\partial \sigma_x}{\partial_x} + \frac{\partial \tau_{xy}}{\partial_y} + \frac{\partial \tau_{xz}}{\partial_z} + X = 0 \\ \frac{\partial \sigma_y}{\partial_y} + \frac{\partial \tau_{xy}}{\partial_x} + \frac{\partial \tau_{yz}}{\partial_z} + Y = 0 \\ \frac{\partial \sigma_z}{\partial_z} + \frac{\partial \tau_{xz}}{\partial_x} + \frac{\partial \tau_{yz}}{\partial_y} + Z = 0 \end{aligned} \tag{13}$$

and the boundary conditions

$$\bar{X} = \sigma_x l + \tau_{xy} m + \tau_{xz} n$$
$$\bar{Y} = \sigma_y m + \tau_{yz} n + \tau_{xy} l \qquad (14)$$
$$\bar{Z} = \sigma_z n + \tau_{xz} l + \tau_{yz} m$$

then from the uniqueness of solutions of the elasticity equations it follows that the assumptions made at the start are correct and the solution obtained is the exact solution of the torsion problem.

Consider a prismatic bar of any cross-section twisted by couples applied at the ends, Fig. 10.

Guided by the solution for a circular shaft, Saint-Venant assumes that the deformation of the twisted shaft consists (where θ is the angle of rotation)

$$\tau = G\theta r \qquad (15)$$

of rotations of cross-sections of the shaft as in the case of a circular shaft and

$$\tau_{yz} = G\theta r\frac{x}{r} = G\theta x$$
$$\tau_{xz} = -G\theta r\frac{y}{r} = -G\theta y \qquad (16)$$

of warping of the cross-sections which is the same for all cross-sections. Taking the origin of coordinates in the end cross-section (Fig. 10)

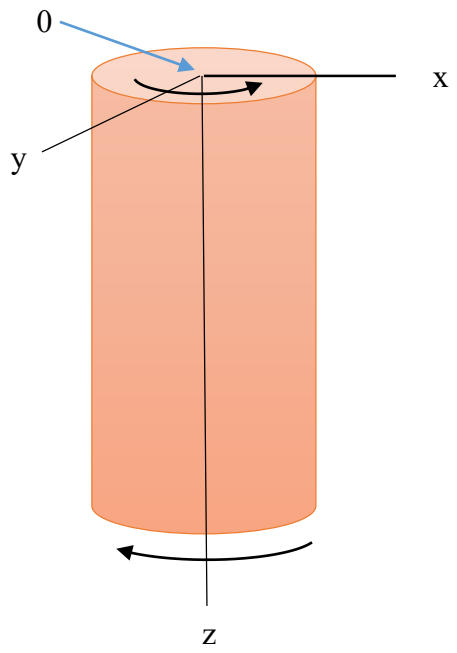


Fig. 10 Free prismatical bar with circle cross-section. **[2]**

we find that the displacements corresponding to rotation of cross-sections are

$$u = -\theta zy, v = \theta zx \tag{17}$$

where θz is the angle of rotation of the cross-section at a distance z from the origin.

The warping of cross-sections is defined by a function

$$\omega = \theta\psi(x, y) \tag{18}$$

With the assumed displacements (17) and (18) we calculate the components of strain from equation

$$\begin{gathered} \varepsilon_x = \frac{du}{dx}, \varepsilon_y = \frac{dv}{dy}, \varepsilon_z = \frac{d\omega}{dz} \\ \gamma_{xy} = \frac{du}{dy} + \frac{dv}{dx}, \gamma_{xz} = \frac{du}{dz} + \frac{d\omega}{dx}, \gamma_{yz} = \frac{dv}{dz} + \frac{d\omega}{dy} \end{gathered} \tag{19}$$

which give

$$\begin{gathered} \varepsilon_x = \varepsilon_y = \varepsilon_z = \gamma_{xy} = 0 \\ \gamma_{xz} = \frac{\partial\omega}{\partial x} + \frac{\partial u}{\partial z} = \theta\left(\frac{\partial\psi}{\partial x} - y\right) \\ \gamma_{yz} = \frac{\partial\omega}{\partial y} + \frac{\partial v}{\partial z} = \theta\left(\frac{\partial\psi}{\partial y} + x\right) \end{gathered} \tag{20}$$

the corresponding components of stress, from

$$\begin{aligned} \varepsilon_x &= \frac{1}{E}\left[\sigma_x - v\left(\sigma_y + \sigma_z\right)\right] \\ \varepsilon_y &= \frac{1}{E}\left[\sigma_y - v(\sigma_x + \sigma_z)\right] \\ \varepsilon_z &= \frac{1}{E}\left[\sigma_z - v\left(\sigma_x + \sigma_y\right)\right] \end{aligned} \tag{21}$$

$$\gamma_{xy} = \frac{1}{G}\tau_{xy}, \gamma_{yz} = \frac{1}{G}\tau_{yz}, \gamma_{zx} = \frac{1}{G}\tau_{zx} \tag{22}$$

Eqs. (21) and (22), are

$$\sigma_x = \sigma_y = \sigma_z = \tau_{xy} = 0$$
$$\tau_{xz} = G\theta\left(\frac{\partial\psi}{\partial x} - y\right)$$
$$\tau_{yz} = G\theta\left(\frac{\partial\psi}{\partial y} + x\right) \tag{23}$$

It can be seen that with the assumptions (17) and (18) regarding the deformation, there will be no normal stresses acting between the longitudinal fibers of the shaft or in the longitudinal direction of those fibers. There also will be no distortion in the planes of cross-sections, since $\varepsilon_x, \varepsilon_y, \gamma_{yz}$ vanish. We have at each point pure shear, defined by the components τ_{xz}andτ_{yz}. The function $\psi(x, y)$, defining warping of cross-section, must now be determined in such a way that equations of equilibrium (13) will be satisfied. Substituting expressions (23) in these equations and neglecting body forces we find that the function ψ must satisfy the equation

$$\frac{\partial^2\psi}{\partial x^2} + \frac{\partial^2\psi}{\partial y^2} = 0 \tag{24}$$

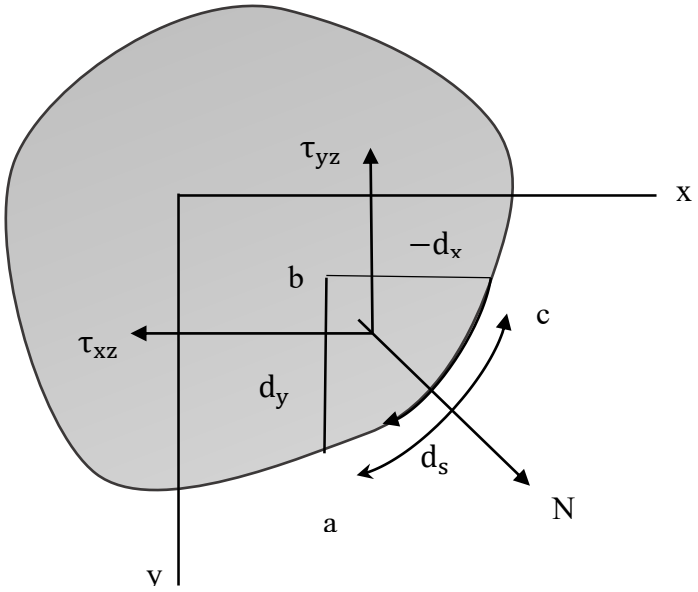


Fig. 11 The resultant shearing stress at the boundary is directed along the tangent to the boundary. **[2]**

Consider now the boundary conditions (14). For the lateral surface of the bar, which is free from external forces and has normals perpendicular to the z-axis, we have $\overline{X} = \overline{Y} = \overline{Z} = 0$ and cos (Nz) =n=0. The first two of Eqs. (14) are identically satisfied and the third gives

$$\tau_{xz} l + \tau_{yz} m = 0 \tag{25}$$

which means that the resultant shearing stress at the boundary is directed along the tangent to the boundary, Fig. 11. It was shown before (see page 1 about Torsion of prismatical bars) that this condition must be satisfied if the lateral surface of the bar is free from external forces.

Considering an infinitesimal element abc at the boundary and assuming that s is increasing in the direction from c to a, we have

$$l = \cos(Nx) = \frac{d_y}{d_s}, m = \cos(Ny) = -\frac{d_x}{d_s}$$

and Eq. (25) becomes

$$\left(\frac{\partial \psi}{\partial x} - y\right)\frac{dy}{ds} - \left(\frac{\partial \psi}{\partial y} + x\right)\frac{dx}{ds} = 0 \tag{26}$$

Thus each problem of torsion is reduced to the problem of finding a function ψ satisfying Eq. (24) and the boundary condition (26).

An alternative procedure, which has the advantage of leading to a simpler boundary condition, is as follows. In view of the vanishing of $\sigma_x, \sigma_y, \sigma_z, \tau_{xy}$ [Eqs. (23)] the equations of equilibrium (13) reduce to

$$\frac{\partial \tau_{xz}}{\partial z} = 0, \frac{\partial \tau_{yz}}{\partial z} = 0, \frac{\partial \tau_{xz}}{\partial x} + \frac{\partial \tau_{yz}}{\partial y} = 0$$

the first two are already satisfied since τ_{xz} ans τ_{yz}, as given by Eqs. (23), are independent of z. The third means that we can express τ_{xz} and τ_{yz} as

$$\tau_{xz} = \frac{\partial \varphi}{\partial y}, \quad \tau_{yz} = -\frac{\partial \varphi}{\partial x} \tag{27}$$

where φ is a function of x and y, called the stress function.

From Eqs.(23) and (27) we have

$$\frac{\partial\varphi}{\partial y} = G\theta\left(\frac{\partial\psi}{\partial x} - y\right), \quad -\frac{\partial\varphi}{\partial x} = G\theta\left(\frac{\partial\psi}{\partial y} + x\right) \tag{28}$$

Eliminating ψ by differentiating the first respect to y, the second with respect to x, and subtracting from the first, we find that the stress function must satisfy the differential equation

$$\frac{\partial^2\varphi}{\partial x^2} + \frac{\partial^2\varphi}{\partial y^2} = F \tag{29}$$

where

$$F = -2G\theta. \tag{30}$$

The boundary condition (25) becomes, introducing Eq. (27)

$$\frac{\partial\varphi}{\partial y}\frac{dy}{ds} + \frac{\partial\varphi}{\partial x}\frac{dx}{ds} = \frac{d\varphi}{ds} = 0. \tag{31}$$

This shows that the stress function φ must be constant along the boundary of the cross-section. In the case of singly connected boundaries, e.g., for solid bars, this constant can be chosen arbitrarily, and in the following discussion we shall take it equal to zero. Thus the determination of the stress distribution over a cross-section of a twisted bar consists in finding of the function φ which satisfies Eq. (29) and is zero at the boundary.

Let us consider now the conditions at the ends of the twisted bar. The normal to the end cross-sections are parallel to the z-axis. Hence $l = m = 0, n = \pm 1$ and Eq. (14) becomes

$$\overline{X} = \pm\tau_{xz}, \qquad \overline{Y} = \pm\tau_{yz} \tag{32}$$

in which the + sign should be taken for the end of the bar for which the external normal has the direction of the positive z-axis, as for the lower end of the bar in Fig. 10. We see that over the ends the shearing forces are distributed in the same manner as the shearing stresses over the cross-sections on the bar. It is easy to prove that these forces give us a torque. Substituting in Eqs, (32) from (27) and observing that φ at the boundary is zero, we find

$$\iint \overline{X}\,dxdy = \iint \tau_{xz}\,dxdy = \iint \frac{d\varphi}{dy}dxdy = \int dx \int \frac{d\varphi}{dy}dy = 0$$
$$\iint \overline{Y}\,dxdy = \iint \tau_{yz}\,dxdy = -\iint \frac{d\varphi}{dx}dxdy = -\int dy \int \frac{d\varphi}{dx}dx = 0$$

Thus the resultant of the forces distributed over the ends of the bar is zero, and these forces represent a couple the magnitude of which is

$$M_t = \iint (\bar{Y}x - \bar{X}y)\,dx\,dy = -\iint \frac{d\varphi}{dx}x\,dx\,dy - \iint \frac{d\varphi}{dy}ydxdy. \tag{33}$$

Integrating this by parts, and observing that $\varphi = 0$ at the boundary, we find

$$M_t = 2\iint \varphi\,dx\,dy \tag{34}$$

each of the integrals in the last member of Eq. (33) contributing one half of this torque. Thus we find that half the torque is due to the stress component τ_{xz} and the other half to τ_{yz}.

We see that by assuming the displacements (17) and (18), and determining the stress components τ_{xz}, τ_{yz} from Eqs. (29), (30) and (31), we obtain a stress distribution which satisfies the equations of equilibrium (13), leaves the lateral surface of the bar free from external forces, and sets up at the ends the torque given by Eq. (34). The compatibility conditions

$$\begin{aligned}
(1+v)\nabla^2\sigma_x + \frac{d^2\theta}{dx^2} = 0, \qquad & (1+v)\nabla^2\tau_{yz} + \frac{d^2\theta}{dydz} = 0 \\
(1+v)\nabla^2\sigma_y + \frac{d^2\theta}{dy^2} = 0, \qquad & (1+v)\nabla^2\tau_{xz} + \frac{d^2\theta}{dxdz} = 0 \\
(1+v)\nabla^2\sigma_z + \frac{d^2\theta}{dz^2} = 0, \qquad & (1+v)\nabla^2\tau_{xy} + \frac{d^2\theta}{dxdy} = 0
\end{aligned} \tag{35}$$

need not be considered since the stress has been derived from the displacements

$$\frac{\partial^2 u}{\partial x^2} = \frac{\partial \varepsilon_x}{dx}, \qquad \frac{\partial^2 u}{\partial y^2} = \frac{\partial \gamma_{xy}}{dy} - \frac{\partial \varepsilon_y}{dx}, \qquad \frac{\partial^2 u}{\partial z^2} = \frac{\partial \gamma_{xz}}{dz} - \frac{\partial \varepsilon_x}{dx}$$

$$\frac{\partial^2 u}{\partial x dy} = \frac{\partial \varepsilon_x}{dy}, \qquad \frac{\partial^2 u}{\partial x\,dz} = \frac{\partial \varepsilon_x}{dz}, \qquad \frac{\partial^2 u}{\partial y\,dz} = \frac{1}{2}\left(\frac{\partial \gamma_{xz}}{dy} + \frac{\partial \gamma_{xy}}{dz} - \frac{\partial \gamma_{yz}}{dx}\right)$$

and

$$\begin{aligned}
u' &= a + by - ez \\
v' &= d - bx + ez \\
\omega' &= f + ex - ey.
\end{aligned}$$

Thus all the equations of elasticity are satisfied and the solution obtained in this manner is the exact solution of the torsion problem.

It was pointed out that the solution requires that the forces at the ends of the bar should be distributed in a definite manner. But the practical application of the solution is not limited to such success. From Saint-Venant's principle it follows that in a long twisted bar, at a sufficient distance from the ends, the stresses depend only on the magnitude of the torque T and are practically independent of the manner in which the tractions are distributed over the ends. More details you can find in [2].

2.2.1. Torsion of rectangular bars

Rectangular profiles are very important in engineering practice. An analytical solution of torsion will be solved for square and rectangular bars. We shell consider a rectangle $\Omega = (-\frac{a}{2}, \frac{a}{2}) \cdot (-\frac{b}{2}, \frac{b}{2})$ with $a \geq b > 0$. The Airy stress function will be looked for in form of a double sum of functions having zero values at the boundary Γ. To this purpose we choose products of $\cos\left(k\,\pi\,\frac{y}{a}\right)$ and $\cos\left(l\,\pi\,\frac{z}{b}\right)$ with positive odd integers k, l since they are zero at Γ and have "good" derivatives. Let us denote the set of positive odd integers by $N_{odd} = \{1, 3, 5, 7, \dots\}$ and put

$$\Phi(y, z) = \sum_{k \in N_{odd}} \sum_{l \in N_{odd}} c_{kl} \cos\frac{k\,\pi\,y}{a} \cos\frac{l\,\pi\,z}{b} \tag{36}$$

with some coefficients c_{kl}. The double sum is supposed to satisfy the equation: $\Delta\Phi \equiv \frac{\partial^2\Phi}{\partial y^2} + \frac{\partial^2\Phi}{\partial z^2} = -2$. Inserting Φ into left hand side of this equation, we obtain

$$\Delta\Phi(y, z) = \sum_{k \in N_{odd}} \sum_{l \in N_{odd}} -c_{kl} \left[\frac{k^2\pi^2}{a^2} + \frac{l^2\pi^2}{b^2}\right] \cos\frac{k\,\pi\,y}{b} \cos\frac{l\,\pi\,z}{h} = -2 \tag{37}$$

to find the coefficients c_{kl} we express -2 by means of the series of the same type. Let us expand the even function $f(y) = 1$ on $(-\frac{a}{2}, \frac{a}{2})$ into a cosine series

$$\sum_{k\in N_{odd}} c_k \cos\frac{k\pi y}{a}.$$

since $\{\cos\left(\frac{k\pi y}{a}\right), k \in N_{odd}\}$ form a complete orthogonal sequence in the Hilbert space of even square integrable functions on $(-\frac{a}{2},\frac{a}{2})$, the coefficients c_k with odd k are given by

$$c_k = \frac{2}{\frac{a}{2}}\int_0^{\frac{a}{2}} f(y)\cos\frac{k\pi y}{a}dy = \frac{4}{a}\left[\frac{a}{k\pi}\sin\frac{k\pi y}{a}\right]_0^{\frac{a}{2}} = (-1)^{\frac{k-1}{2}}\frac{4}{k\pi}.$$

In the same way we expand the function $g(z) = 1$ on $\left(-\frac{b}{2},\frac{b}{2}\right)$. Thus we obtained

$$\sum_{k\in N_{odd}} (-1)^{\frac{k-1}{2}}\frac{4}{k\pi}\cos\frac{k\pi y}{a} = 1,$$

$$\sum_{l\in N_{odd}} (-1)^{\frac{l-1}{2}}\frac{4}{l\pi}\cos\frac{l\,\pi\, z}{b} = 1.$$

Multiplying product of both series with constant -2 we obtain equality

$$\sum_{k\in N_{odd}}\sum_{l\in N_{odd}} -2(-1)^{\frac{k+l}{2}-1}\frac{4^2}{kl\pi^2}\cos\frac{k\pi y}{a}\cos\frac{l\,\pi\, z}{b} = -2. \qquad (38)$$

Since both series in (37) and (38) are of the same type and have the same sum, the corresponding coefficient must be equal. Comparing both series we obtain formula for the coefficients c_{kl}:

$$c_{kl} = 2\frac{4^2}{kl\pi^2}(-1)^{\frac{k+l-2}{2}}\left[\frac{k^2\pi^2}{a^2}+\frac{l^2\pi^2}{b^2}\right]^{-1} = \frac{2^5a^2b^2}{\pi^4}\frac{(-1)^{\frac{k+l}{2}-1}}{kl(k^2b^2+l^2a^2)}.$$

Thus the Airy stress function equals to

$$\Phi(y,z) = \frac{2^5 a^2 b^2}{\pi^4} \sum_{k\in N_{odd}} \sum_{l\in N_{odd}} \frac{(-1)^{\frac{k+l}{2}-1}}{kl(k^2b^2+l^2a^2)} \cos\frac{k\pi y}{a} \cos\frac{l\pi z}{b}. \qquad (39)$$

We obtained result in the form of an infinite series. Due to estimate

$$|\, c_{kl} \cos\frac{k\pi y}{a} \cos\frac{l\pi z}{b}| \le |c_{kl}| \le const.\frac{1}{k^2 l^2},$$

the series converge uniformly, since $\sum_{k\in N_{odd}} \sum_{l\in N_{odd}} \frac{1}{k^2 l^2}$ equals to product of two series $(\sum_{k\in N_{odd}} \frac{1}{k^2})$ $(\sum_{l\in N_{odd}} \frac{1}{l^2})$ which both have finite sum. Let us compute the moment J. Since

$$\int_{-a/2}^{a/2} \cos\left(\frac{k\pi y}{a}\right) dy = \frac{2a}{k\pi}(-1)^{\frac{k-1}{2}}, \qquad \int_{-b/2}^{b/2} \cos\left(\frac{l\pi z}{b}\right) dz = \frac{2b}{l\pi}(-1)^{\frac{l-1}{2}}$$

the formula $J = 2\iint_\Omega \Phi(y,z)dydz$ yields

$$J = 2^8 \frac{a^3 b^3}{\pi^6} \sum_{k\in N_{odd}} \sum_{l\in N_{odd}} \frac{1}{k^2 l^2 (k^2 b^2 + l^2 a^2)}.$$

Let us denote the ratio $r = \frac{a}{b}$. Using a function $K_1(r)$ the relation can be written to

$$J = K_1\left(\frac{a}{b}\right) ab^3,$$

r	1	1.5	2	3	4	5	6	8	10	∞
$K_1(r)$	0.141	0.196	0.229	0.263	0.281	0.291	0.298	0.307	0.312	1/3
$K_2(r)$	0.208	0.231	0.246	0.267	0.282	0.292	0.299	0.307	0.313	1/3

Tab. 1 Numerical values of $\mathbf{K_1(r)}$ **and** $\mathbf{K_2(r)}$**. [3]**

where the dimensionless function $K_1(r)$ depending on the ratio r is given by

$$K_1(r) = \frac{2^8}{\pi^6} \sum_{k \in N_{odd}} \sum_{l \in N_{odd}} \frac{r^2}{k^2 l^2 (k^2 + r^2 l^2)}.$$

The values of $K_1(r)$ for some ratios r are in Tab. 1.

Relations

$$\tau_{xy} = \alpha\mu \frac{\partial \Phi(y,z)}{\partial z}, \tau_{xz} = -\alpha\mu \frac{\partial \Phi(y,z)}{\partial y} \tag{40}$$

yields the nonzero components of the stress tensor

$$\tau_{xy}(y,z) = \alpha\mu \frac{2^5 a^2 b}{\pi^3} \sum_{k \in N_{odd}} \sum_{l \in N_{odd}} \frac{(-1)^{\frac{k+l}{2}}}{k(k^2 b^2 + l^2 a^2)} \cos\frac{k\pi y}{a} \sin\frac{l\pi z}{b},$$

$$\tau_{xz}(y,z) = -\alpha\mu \frac{2^5 a\, b^2}{\pi^3} \sum_{k \in N_{odd}} \sum_{l \in N_{odd}} \frac{(-1)^{\frac{k+l}{2}}}{l(k^2 b^2 + l^2 a^2)} \sin\frac{k\pi y}{a} \cos\frac{l\pi z}{b}.$$

According to the remark in the end of Section 2, the maximum of the stress in the middle point $[0, \pm\frac{b}{2}]$ of the longer side of the rectangle. From Eq. (40) simple computation yields

$$|T|_{max} = \alpha\mu \frac{2^5 a^2}{\pi^3 b} \sum_{k \in N_{odd}} \sum_{l \in N_{odd}} \frac{(-1)^{\frac{k-1}{2}}}{k(k^2 + r^2 l^2)}.$$

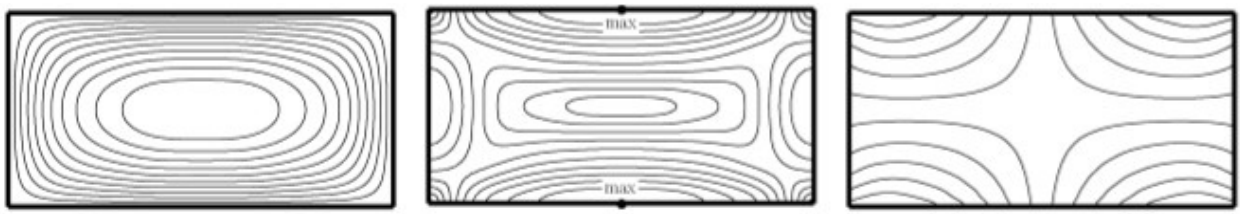
Using a function $K_2(r)$ the relation can be written to

$$|T|_{max} = \alpha\mu a b^2 K_2\left(\frac{a}{b}\right),$$

where the dimensionless function $K_2(r)$ depending on the ratio r=a/b and $K_1(r)$ is given by

$$K_2(r) = K_1(r) \frac{\pi^3}{2^5 r^2} \left[\sum_{k \in N_{odd}} \sum_{l \in N_{odd}} \frac{(-1)^{\frac{k-1}{2}}}{k(k^2 + r^2 l^2)} \right]^{-1}. \tag{41}$$

Some numerical values of $K_2(r)$ are listed in the Table 1. In mechanics the $|T|_{max}$ is often expressed in the form $|T|_{max} = \frac{M}{W}$, where the section twisting moment W is given by $W = ab^2K_2(a/b)$.



(a) (b) (c)

Fig. 12 Results for a rectangular profile with $\boldsymbol{\mu = 1,\ \alpha = 1, a = 2}$ **and** $\boldsymbol{b = 1}$: (a) stress function Φ, (b) modulus of the shear stress $\boldsymbol{\tau}$; the maximum values of $\boldsymbol{\tau}$ occur in the middle of longer sides, (c) deflection of the cross-section. **[3]**

The deflection φ is given by its differential. Using $\frac{\partial\varphi}{\partial y} = \frac{\partial\Phi}{\partial z} + z,\ \frac{\partial\varphi}{\partial z} = -\frac{\partial\Phi}{\partial y} - y$ we obtain

$$\varphi(y,z) = \frac{2^5\, a^3 b}{\pi^4} \sum_{k\in N_{odd}} \sum_{l\in N_{odd}} \frac{(-1)^{\frac{k+l}{2}}}{k^2(k^2b^2 + l^2a^2)} \sin\frac{k\,\pi\, y}{a} \sin\frac{l\,\pi\, z}{b} + yz\,,$$

its values are also plotted in the Fig. 12c. More details you can find in [3].

2.2.2. Torsion of triangular bars

Third order polynomials enable to solve the case of triangle profile. Let us consider a general triangle Ω with one side being parallel to the axis y, i.e. a subset of the line z=-c. Let the other sides lie on the lines $z = a_1y + b_1$ and $z = a_2y + b_2$, where $a_1, a_2, b_1, b_2 \in R$, and $c > 0$. Then the level curve $P_3(y,z)=0$ of the polynomial

$$P_3(y,z) = k(a_1y + b_1 - z)(\,a_2y + b_2 - z)(z + c)$$

With $k \neq 0$ bounds the triangle Ω. Simple calculation yields

$$\Delta P_3(y,z) = -2k(a_1 + a_2)y + 2k(\,a_1a_2 + 3)z + 2k(a_1a_2c - b_1 - b_2 + c)$$

The right-hand side is constant if the coefficients by terms y and z are zero, i.e. if $a_1 + a_2 = 0$ and $a_1 a_2 + 3 = 0$, which implies $a_1 = -a_2 = \pm\sqrt{3}$. Thus the triangle Ω must be equilateral.

Choosing $b_1 = b_2 = b$ and b=2c the triangle has vertices $\left[-\sqrt{3c}, -c\right], \left[\sqrt{3c}, -c\right], [0,2c]$ with its center of gravity in [0, 0].

The polynomial P_3 is a solution to the problem $M = 2a\mu \iint_\Omega \Phi dydz$ if the right-hand constant equals to -2, i.e. $-k(a_1 a_2 c - b_1 - b_2 + c) = k6c = 1$ which yields $k = (6c)^{-1}$. We obtained the Airy stress function

$$\Phi(y,z) = \frac{1}{6c}\left(\sqrt{3y} + 2c - z\right)\left(-\sqrt{3y} + 2c - z\right)(z + c). \tag{42}$$

The moment J of the cross-section can be computed using, $J = 2\iint_\Omega \Phi(y,z)dydz$:

$$J = 2\frac{1}{6c}\int_{-c}^{2c}\left(\int_{(z-2c)/\sqrt{3}}^{(2c-z)/\sqrt{3}} \Phi(y,z)dy\right)dz = \frac{9\sqrt{3}}{5}c^4 \tag{43}$$

The only non-zero components of the stress tensor are

$$\tau_{xy} = -\alpha\mu\frac{y^2 - z^2 + 2cz}{2c},\ \tau_{xy} = \alpha\mu\frac{y(z+c)}{c}. \tag{44}$$

The modulus of the shear stress

$$|\tau| = \sqrt{\tau_{xy}^2 + \tau_{xz}^2} = \alpha\mu\frac{1}{2c}\sqrt{(y^2 - z^2 + 2cz)^2 + 4y^2(z+c)^2} \tag{45}$$

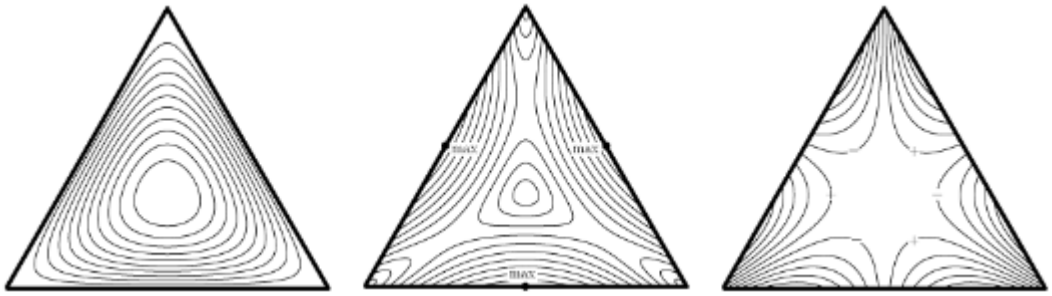
is plotted of the figure 13b, maximum values $|T|_{max} = \alpha\mu 3c/2$ are attained in the center points of the sides, e.g. in [0, -c]. The twist section modulus is $W = 6\sqrt{3}\,c^2/5$. Concerning the deflection φ, in our case the equations $\frac{\partial\varphi}{\partial y} = \frac{\partial\Phi}{\partial z} + z$, $\frac{\partial\varphi}{\partial z} = -\frac{\partial\Phi}{\partial y} - y$ read

$$\frac{\partial\varphi}{\partial y} = \frac{z^2 - 2zc - y^2}{2c} + z, \qquad \frac{\partial\varphi}{\partial z} = \frac{(z+c)y}{c} - y.$$

Standard computation with $\iint_\Omega \varphi(y,z) = 0$ yields the deflection function φ

$$\varphi(y,z) = \frac{1}{6c}y(3z^2 - y^2), \tag{46}$$

which is also plotted on Fig. 13c. More details you can find in [3].



(a) (b) (c)

Fig. 13 Triangular cross-section with $\boldsymbol{\mu = 1, \alpha = 1 \text{ and } c = 1}$: (a) stress function Φ, (b) modulus of the shear stress $\boldsymbol{\tau}$, the maximum values of |T| occur in the middle of the sides, (c) deflection of the cross-section. **[3]**

3. Finite Elements Method – deformation variant

Finite Element Analysis was first developed in the early 1960's as a simulation and design tool in the aerospace and nuclear industries where the safety of structures is critical. The process starts with the creation of a geometric model. Then, the model is subdivided (meshed) into small pieces (elements) of simple shapes connected at specific node points. Within each element, the variation of displacement is assumed to be determined by simple polynomial shape functions and nodal displacements. Equations for the strains and stresses are developed in terms of the unknown nodal displacements. From this, the equations of equilibrium are assembled in a matrix which can be easily programmed and solved on a computer. After applying the appropriate boundary conditions, the nodal displacements are found by solving the matrix stiffness equation. Once the nodal displacements are known, element stresses and strains can be calculated. More details you can find in [4].

FEM-**F**inite **E**lement **M**ethod

- ✓ A numerical method
- ✓ Mathematical representation of an actual problem
- ✓ Approximate method

Analysis of properties, states and behavior of technical objects is an important task of Engineering Mechanics. Strain-stress analysis of solid bodies ranks to these problems. Continuum mechanics and especially elasticity theory provides tools for this analysis.

The strain-stress analysis passed through its development, analytical approaches predominating in the past are at present replaced by numerical tools as Finite Element Method, Finite Volume Method, Boundary Element Method, etc. In comparison to the classical methods the numerical methods are universal in sense that their applicability is independent of geometry of the body, material characteristics, etc. This may indicate that the period of analytical elasticity ended. But it

is not the case. There are reasons to keep on using both the analytical methods and the numerical methods in the strain-stress analysis.

The numerical method Finite Element Method enable to compute numerical values of strain-stress state of the particular material in particular points under particular loads. On the other hand they provide no formulas which can be used for predicting the change of the values under changing load, size, stiffness etc. The analytical methods yielding formulas enable this prediction. Unfortunately they can be applied only to special shapes of bodies and loads.

The Finite Element Method makes calculations at a limited (Finite) number of points and then interpolates the results for the entire domain (surface or volume).
Finite-Any continuous object has infinite degrees of freedom and it is not possible to solve the problem in this format. The Finite Element Method reduces the degrees of freedom from infinite to finite with the help of discretization or meshing (nodes and elements).
Element-All of the calculations are made at a limited number of points known as nodes. The entity joining nodes and forming a specific shape such as quadrilateral or triangular is known as an Element. To get the value of a variable (say displacement) anywhere in between the calculation points, an interpolation function (as per the shape of the element) is used.
Method-There are 3 methods to solve any engineering problem. Finite element analysis belongs to numerical method category. You can see Finite Element methods subsequent in the diagram, Fig. 14. More details you can find in [5].

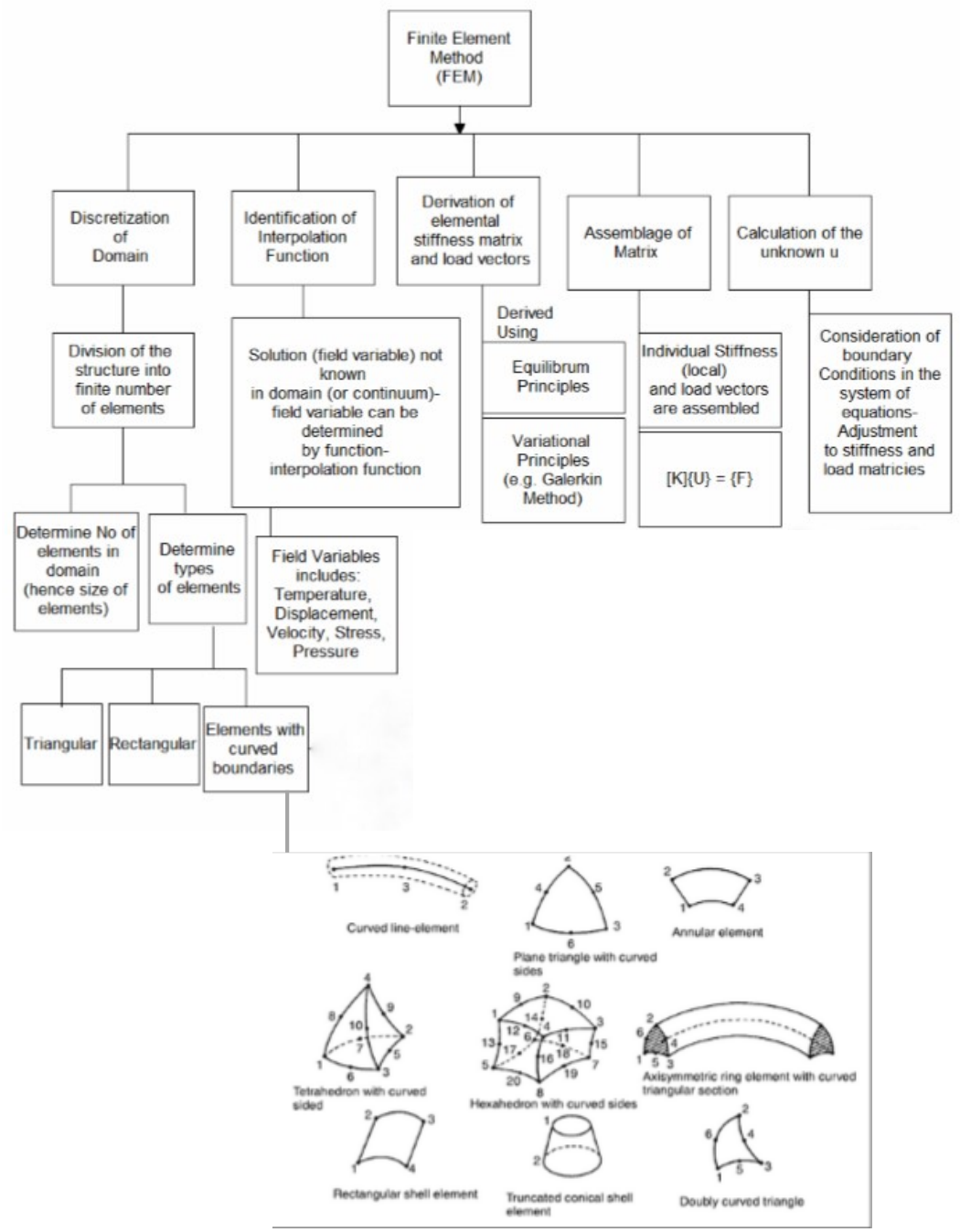


Fig. 14 Flow chart illustrates finite element method. **[5]**

A 2D solid element can have a triangular, rectangular or quadrilateral shape with straight or curved edges.

3.1. Theory and derivation of stiffness matrix and right hand side

Now the partial equation (29) will be solved by FEM. To get the solution $\Phi(x,y)$, we can solve modified equation

$$-\left(\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}\right)=f(x,y)=1 \qquad (48)$$
$$u_\Gamma = 0$$

with RHS f(x,y) equal to 1. So over the whole area (see Fig. 15) a unit volume force acts.

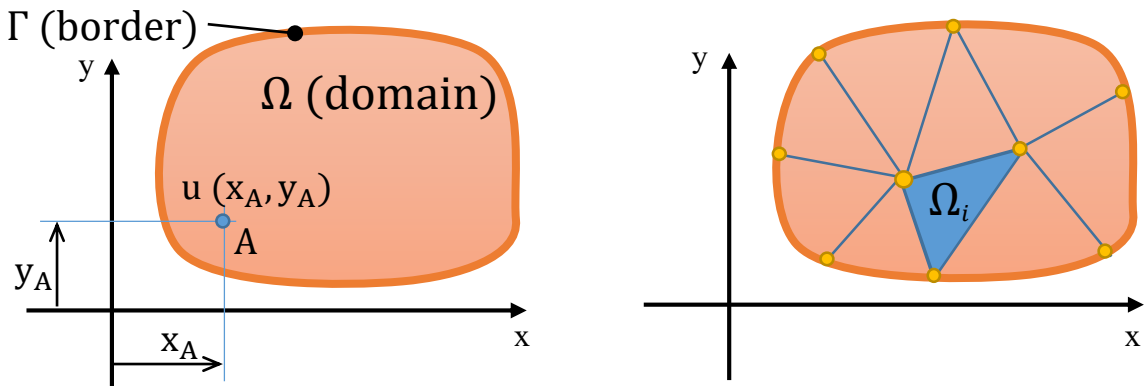


Fig. 15 Solving domain.

The relation between original and new unknowns is

$$\Phi(x,y) = 2G\theta. \qquad (49)$$

The equilibrium equation above (48) can be multiplied by testing function v satisfying the same boundary condition as the function u (v is zero on the border Γ). Then the term is integrated over whole domain Ω

$$\iint_\Omega \frac{\partial v}{\partial x}\frac{\partial u}{\partial x}+\frac{\partial v}{\partial y}\frac{\partial u}{\partial y}\,dxdy = \iint_\Omega vf\,dxdy. \qquad (50)$$

The integrals above can be evaluated separately over smaller subdomains (over Ω_i, see Fig. 15 right) and then they are obtained as summarization of all contributions of Ω_i. Obviously, Ω_i corresponds to i-the finite element. Then the term (50) can be written as

$$\sum_{i=0}^{N-1} \iint_{\Omega_i} \left(\frac{\partial v_h^i}{\partial x}\right)^T \frac{\partial u_h^i}{\partial x} + \left(\frac{\partial v_h^i}{\partial y}\right)^T \frac{\partial u_h^i}{\partial y} \, dxdy = \sum_{i=0}^{N-1} \iint_{\Omega_i} (v_h^i)^T \, f \, dxdy \qquad (51)$$

where u_h and v_h stem from original functions u and v. The subindex h is pointing out, these functions are assembled separately over each element.

3.2. Triangular element - stiffness matrix, right hand side

Now here will be calculated a stiffness matrix and right hand side (RHS) vector of the triangular element. First, the element stiffness matrices will be calculated, and then the global stiffness matrix of the entire structure will be assembled.
First, evaluation of u will be described. It is considered a linear approximation of the displacements in the form

$$u_h^i(x, y) = a_0 + a_1 \cdot x + a_2 \cdot y. \qquad (52)$$

To get unknown constant a_i over each element, in the term (51) x and y is substituted by vertex coordinates. (Fig. 16)

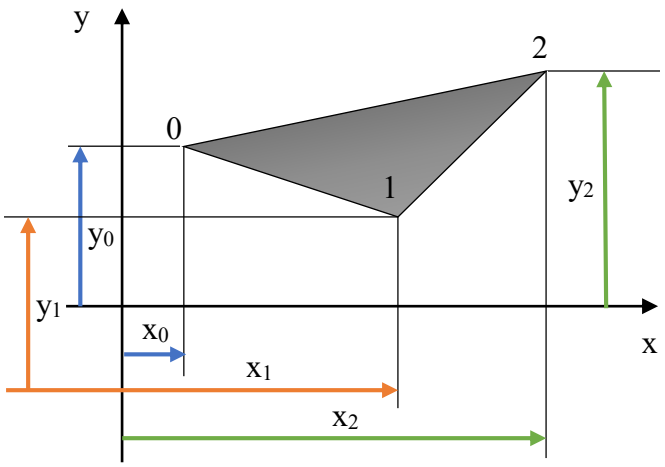


Fig. 16 Triangular element with vertex coordinates.

So for, e.g., first node it is obtained $u(x_0,y_0)=r_0$. Similarly for second and third node written together it is

$$r_h^i = \begin{bmatrix} r_0 \\ r_1 \\ r_2 \end{bmatrix}; \qquad r_h^i = \begin{bmatrix} 1 & x_0 & y_0 \\ 1 & x_1 & y_1 \\ 1 & x_2 & y_2 \end{bmatrix} \begin{pmatrix} a_0 \\ a_1 \\ a_2 \end{pmatrix} = A_h \cdot a_h = r_h^i \Rightarrow\!\!\gg a_h = A_h^{-1} \cdot r_h^i. \tag{53}$$

Then the function u describing an approximation of displacements over the element is given by

$$u_h^i(x,y) = [1|x|y] \begin{bmatrix} a_0 \\ a_1 \\ a_2 \end{bmatrix} = M_h \cdot a_h = (M_h \cdot A_h^{-1}) \cdot r_h^i \tag{54}$$

In literature, the product $N_h = M_h \cdot A_h^{-1}$ is knowns as matrix of shape functions. Then it can be written in the form

$$u_h^i(x,y) = [N_0|N_1|N_2] \begin{bmatrix} r_0 \\ r_1 \\ r_2 \end{bmatrix} = N_h r_h^i. \tag{55}$$

Discretization form of the testing function $v_h(x,y)$ is obtained almost the same way

$$v_h^i(x,y) = [N_0|N_1|N_2] \begin{bmatrix} \delta r_0 \\ \delta r_1 \\ \delta r_2 \end{bmatrix} = N_h \delta r_h^i \tag{56}$$

instead of the displacements r_h^i, there are used virtual displacements δr_h^i .

But in the left integral

$$\iint_\Omega \left(\frac{\partial v_h^i}{\partial x}\right)^T \frac{\partial u_h^i}{\partial x} + \left(\frac{\partial v_h^i}{\partial y}\right)^T \frac{\partial u_h^i}{\partial y} = \iint_\Omega v_h^i f \, dxdy. \tag{57}$$

there are the derivative of u_h^i and v_h^i.

Derivative of u_h according to x is given by

$$\frac{\partial u_h^i}{\partial x} = \frac{\partial N_h}{\partial x} \cdot r_h^i = \frac{\partial M_h}{\partial x} A_h^{-1} \cdot r_h^i = G_{h,x} \cdot r_h^i, \tag{58}$$

where

$$dM_{h,x} = [0 \;\; 1 \;\; 0] = \frac{\partial M_h}{\partial x}, \qquad G_{h,x} = \frac{\partial M_h}{\partial x} A_h^{-1}. \tag{59}$$

The derivation according to y is

$$\frac{\partial u_h^i}{\partial y} = G_{h,y} \cdot r_h^i. \tag{60}$$

where

$$\frac{\partial u_h^i}{\partial y} = \frac{\partial M_h}{\partial y} A_h^{-1} \cdot r_h^i, \qquad G_{h,y} = \frac{\partial M_h}{\partial y} A_h^{-1}. \tag{61}$$

Now the integral in (61) can be replaced by

$$\sum_{i=0}^{N-1} \quad \iint_{\Omega_i} \left(\frac{\partial v_h^i}{\partial x}\right)^T \frac{\partial u_h^i}{\partial x} + \left(\frac{\partial v_h^i}{\partial y}\right)^T \frac{\partial u_h^i}{\partial y} dxdy \tag{62}$$

and the same just for one element

$$\iint_{\Omega_i} \left[\left(\delta r_h^i \right)^T {G_{h,x}}^T G_{h,x} \cdot r_h^i + \left(\delta r_h^i \right)^T {G_{h,y}}^T G_{h,y} \cdot r_h^i \right] dxdy. \tag{63}$$

Because matrices $G_{h,x}$ and $G_{h,y}$ contain only constant entries, the integration is

$$\left(\delta r_h^i\right)^T \left(G_{h,x}{}^T G_{h,x} + G_{h,y}{}^T G_{h,y} \iint \mathrm{dxdy}\right) \cdot r_h^i, \tag{64}$$

where $\iint \mathrm{dxdy}$ is equal to the area Ω_i of the particular element, therefore

$$\left(\delta r_h^i\right)^T \left[G_{h,x}{}^T G_{h,x} + G_{h,y}{}^T G_{h,y}\right] \Omega_i \, r_h^i \tag{65}$$

and finally

$$K_h = \left[G_{h,x}{}^T G_{h,x} + G_{h,y}{}^T G_{h,y}\right] \Omega_i. \tag{66}$$

The integral on the right hand side is written in the form

$$\iint_{\Omega_i} (v_h^i)^T f(x,y)\mathrm{dxdy}, \tag{67}$$

and using the substitution of v_h it changes to

$$\left(\delta r_h^i\right)^T \iint_{\Omega_i} N_h^T f(x,y)\mathrm{dxdy}. \tag{68}$$

The matrix of shape functions will be substituted by already presented form $N_h = M_h \cdot A_h^{-1}$. In term of evaluation of (64), this values to be integrated ($N_h^T f(x,y)$), are not constant. To simplify this process, a numerical integration can be used - Gaussian quadrature with one integrating point. Then the contribution of i-th element to the global RHS is

$$F_h^i = A_h^{-T} \begin{bmatrix} f(x_T, y_T) \\ x_T \cdot f(x_T, y_T) \\ y_T \cdot f(x_T, y_T) \end{bmatrix} \Omega_i, \tag{69}$$

and the derivation is obvious from following part

$$\left(\delta r_h^i\right)^T \iint A_h^{-T} M_h^T f(x,y)dxdy = \left(\delta r_h^i\right)^T \iint A_h^{-T} \begin{bmatrix} 1 \cdot f(x,y) \\ x \cdot f(x,y) \\ y \cdot f(x,y) \end{bmatrix} dxdy \approx$$

$$\left(\delta r_h^i\right)^T A_h^{-T} \begin{bmatrix} f(x_T,y_T) \\ x_T \cdot f(x_T,y) \\ y_T \cdot f(x_T,y_T) \end{bmatrix} \Omega_i = \left(\delta r_h^i\right)^T F_h^i. \quad (70)$$

Because the volume force f is constant (f=1), used numerical integration method provides exact solution in the form

$$F_h^i = A_h^{-T} \begin{bmatrix} f(x_T,y_T) \\ x_T \cdot f(x_T,y) \\ y_T \cdot f(x_T,y_T) \end{bmatrix} \Omega_i.$$

3.3. Global stiffness matrix and global right hand side

For this a node-element connectivity chart (Tab. 2) is created in very simple case. The structure contains just 2 elements (see Fig. 17).

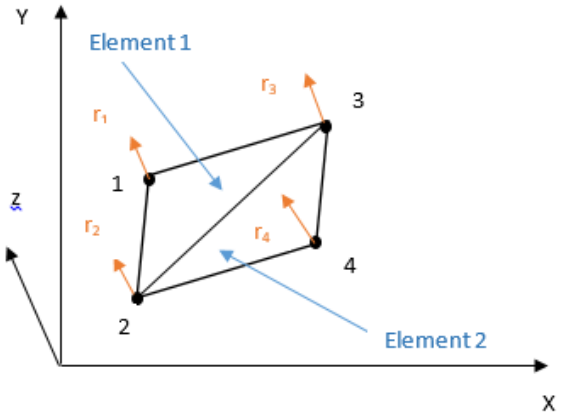


Fig. 17 Two triangular elements.

Element	Node 1	Node 2	Node 3
1	1	2	3
2	2	3	4

Tab. 2 The numbers of nodes the structure with 2 triangular elements.

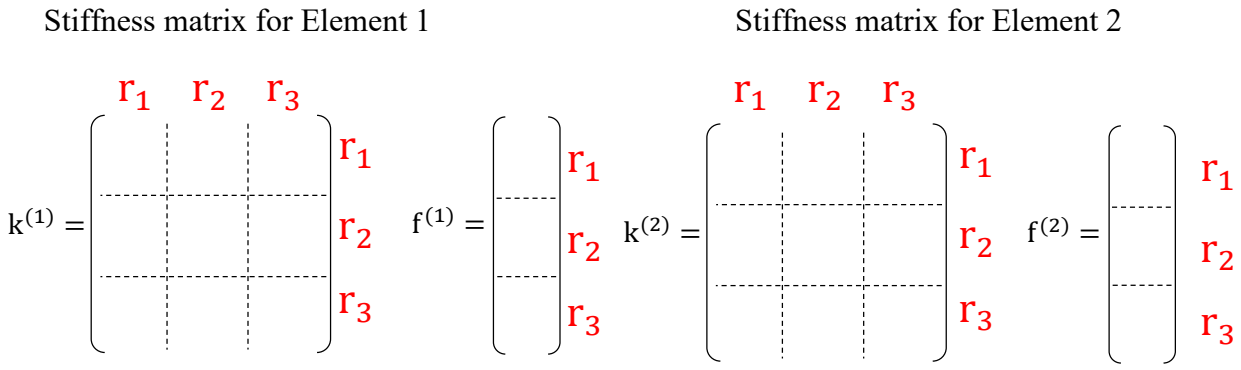


Fig. 18 Stiffness matrixes and right hand side (RHS) vectors of the two triangular element.

Each node has 1 degree of freedom (DOF), so the dimension of matrix K (and RHS F) is equal to the number of nodes. Assembled global stiffness matrix relating to two elements is in Fig. 19.

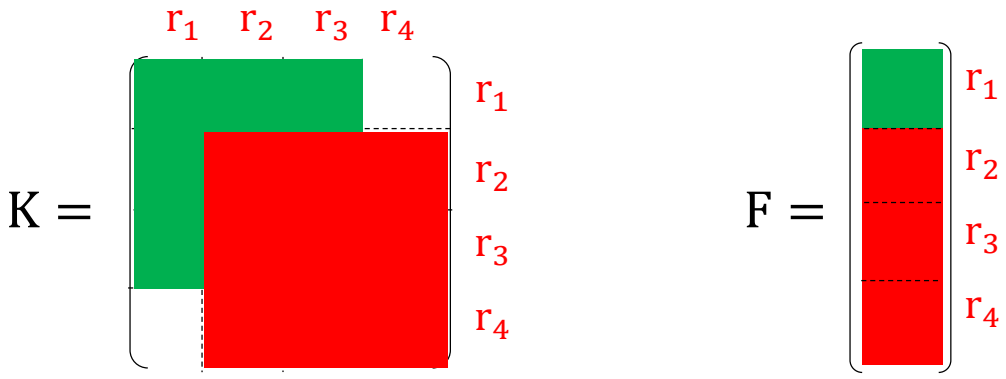


Fig. 19 Global stiffness matrix and RHS.

Similarly, the right-hand side is assembled from contributions of all (here just two) elements. Once, the global stiffness matrix and right-hand-side is assembled, the boundary condition have to be take into account.

3.4. Rectangular element - stiffness matrix, right hand side

Here, the stiffness matrix and RHS for rectangular element will be describe.

First the body (See Fig. 20) was divided into finite elements connected to each other through special points-node

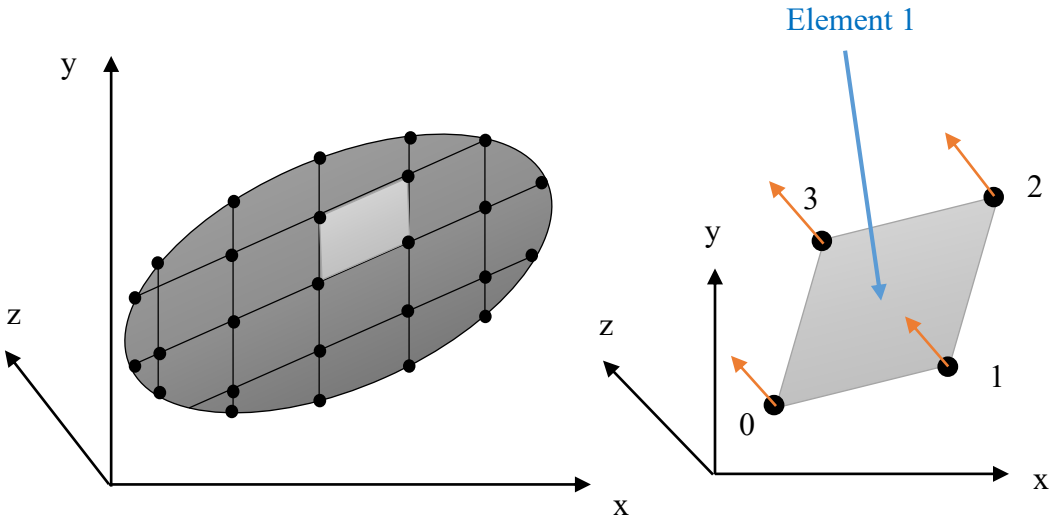


Fig. 20 Rectangular element.

Element	Node 1	Node 2	Node 3	Node 4
1	0	1	2	3

Tab. 3 The number of node.

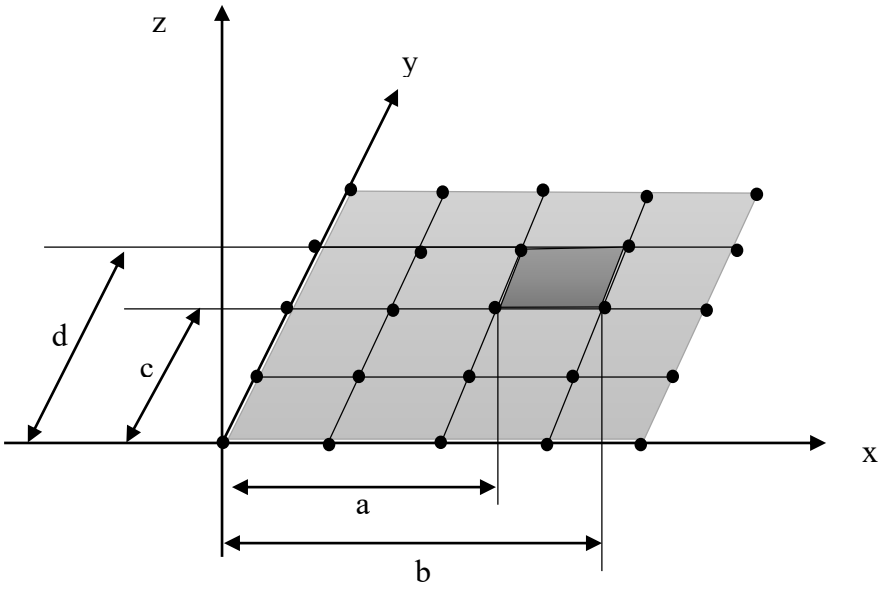


Fig. 21 The integral limits over i-th rectangular element.

Followed derivation is prepared for very special case in which all rectangular elements have parallel edges with axes x and y (see Fig. 22 left). This simplification is used only for square and rectangular cross-section variant.

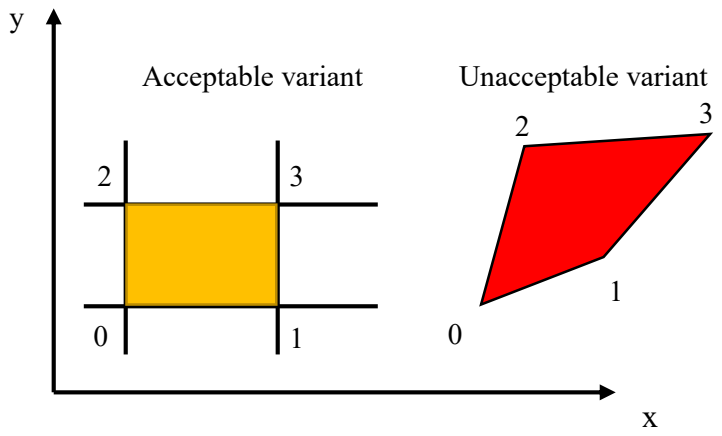


Fig. 22 Rectangular element with parallel edges of x and y axis.

Compared to triangular element, an approximation for the displacements is extended and it reads as

$$u_h^i(x,y) = a_0 + a_1 \cdot x + a_2 \cdot y + a_2 \cdot x \cdot y,$$

so the polynomial matrix will be

$$M_h = [1 \;\; x \;\; y \;\; xy]. \tag{71}$$

Derivative of u_h according to x and y is given, where

$$\frac{\partial M_h}{\partial x} = [\,0 \;\; 1 \;\; 0 \;\; y\,]; \quad \frac{\partial M_h}{\partial y} = [\,0 \;\; 0 \;\; 1 \;\; x\,], \tag{72}$$

and the transformation matrix is

$$A_h = \begin{bmatrix} 1 & x_0 & y_0 & x_0y_0 \\ 1 & x_1 & y_1 & x_1y_1 \\ 1 & x_2 & y_2 & x_2y_2 \\ 1 & x_3 & y_3 & x_3y_3 \end{bmatrix}. \tag{73}$$

Integral for one element representing local stiffness matrix is then

$$k_h = \iint_{\Omega_i} \left(G_{h,x}{}^T G_{h,x} + G_{h,y}{}^T G_{h,y}\right) dxdy. \tag{74}$$

Matrices $G_{h,x}$ and $G_{h,y}$

$$\begin{aligned} G_{h.x} &= \frac{\partial}{\partial x} M_h A_h^{-1} = [\,0 \;\; 1 \;\; 0 \;\; y\,] A_h^{-1}, \\ G_{h.y} &= \frac{\partial}{\partial y} M_h A_h^{-1} = [\,0 \;\; 0 \;\; 1 \;\; x\,] A_h^{-1}, \end{aligned} \tag{75}$$

are not constant. Written in complete form the stiffness matrix is

$$K_h = \iint_{\Omega_i} \left(A_h^{-T} \begin{bmatrix} 0 \\ 1 \\ 0 \\ y \end{bmatrix} [0 \;\; 1 \;\; 0 \;\; y\,] A_h^{-1} + A_h^{-T} \begin{bmatrix} 0 \\ 0 \\ 1 \\ x \end{bmatrix} [0 \;\; 0 \;\; 1 \;\; x\,] A_h^{-1} \right) dxdy, \tag{76}$$

or

$$K_h = \iint_{\Omega_i} A_h^{-T} \left(\begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & y \\ 0 & 0 & 0 & 0 \\ 0 & y & 0 & y^2 \end{bmatrix} + \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & x \\ 0 & 0 & x & x^2 \end{bmatrix} \right) A_h^{-1} \, dxdy. \tag{77}$$

Finally, the expression is

$$K_h = \iint_{\Omega_i} A_h^{-T} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \left(\begin{array}{c|c|c|c} 0 & 0 & 0 & 0 \\ \hline 0 & 1 & 0 & y \\ \hline 0 & 0 & 1 & x \\ \hline 0 & y & x & x^2+y^2 \end{array}\right) \end{array} \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \end{array} A_h^{-1}\, dxdy$$

Fig. 23 Stiffness matrix for one element.

or putting the double integral inside

$$K_h = A_h^{-T} \begin{array}{c} \begin{array}{cccc} 0 & 1 & 2 & 3 \end{array} \\ \left(\begin{array}{c|c|c|c} 0 & 0 & 0 & 0 \\ \hline 0 & \int_c^d \int_a^b dx & 0 & \int_c^d \int_a^b ydxdy \\ \hline 0 & 0 & \int_c^d \int_a^b dx & \int_c^d \int_a^b xdxdy \\ \hline 0 & \int_c^d \int_a^b ydxdy & \int_c^d \int_a^b xdxdy & \int_c^d \int_a^b x^2+y^2 dxdy \end{array}\right) \end{array} \begin{array}{c} 0 \\ 1 \\ 2 \\ 3 \end{array} A_h^{-1}$$

Fig. 24 Global stiffness matrix for one element.

As we can see, elements $K_{h(1,1)}$ and $K_{h(2,2)}$ are equal, so it can be written

$$K_{h(1,1)} = K_{h(2,2)} \int_c^d \int_a^b dxdy = (b-a)(d-c)\,. \tag{78}$$

Due to symmetry, $K_{h(3,1)}$ and $K_{h(1,3)}$ are same as well, so it will be:

$$K_{h(3,1)} = K_{h(1,3)} \int_c^d \int_a^b ydxdy = \frac{1}{2}(d^2-c^2)(b-a) \tag{79}$$

and similarly $K_{h(3,2)} = K_{h(2,3)}$

$$K_{h(3,2)} = K_{h(2,3)} \int_c^d \int_a^b x\,dx\,dy = \frac{1}{2}(b^2 - a^2)(d - c). \tag{80}$$

Finally, the last one $K_{h(3,3)}$ is given by

$$K_{h(3,3)} = \int_c^d \int_a^b x^2 + y^2\,dx\,dy = \frac{1}{3}[(b^3 - a^3)(d - c) + (d^3 - c^3)(b - a)]. \tag{81}$$

Derivation of RHS keeps the same rules already applied onto 3-nodes element:

$$F_h = \int_c^d \int_a^b A_h^{-T}\, M_h^T\, dx\,dy, \tag{82}$$

$$F_h = A_h^{-T} \begin{bmatrix} \iint 1\,dx\,dy \\ \iint x\,dx\,dy \\ \iint y\,dx\,dy \\ \iint xy\,dx\,dy \end{bmatrix}. \tag{83}$$

From Eq. 83 first, second and third integrals are already calculated, the only unknown is: $\iint xy\,dx\,dy$ which will be:

$$F_{h(3)} = \int_c^d \int_a^b xy\,dx\,dy = \frac{1}{4}(b^2 - a^2)\,d^2 - c^2. \tag{84}$$

4. Torsion - numerical solution

4.1. Introduction of own meshgenerator

The Python library provides numerical solutions, and partially, depending on the choses cross-section, analytical solution. It also provides the output parameter which corresponds in some sense to the quadratic moment of area J_ρ. The script calculates tangential stress distribution for all cross-sections.

In the script following types of cross-sections are implemented: square, rectangular, triangular and circle. Also we can solve rectangular cross-section's deformed shape. For them there are followed parameters which can be set-up by the user:

- Lx - length of x-axis,
- Ly - length of y-axis,
- nx - number of elements on the x-axis,
- ny - number of elements on the y-axis,
- n - number of elements for circular and triangular cross-section,
- R - radius,
- D - diameter,
- k - ratio,
- h - height.

5. Validation of numerical results obtained by Python library

Verification of the Pyhton library, all cross-section in Fig. 25 are compared to the analytical terms.

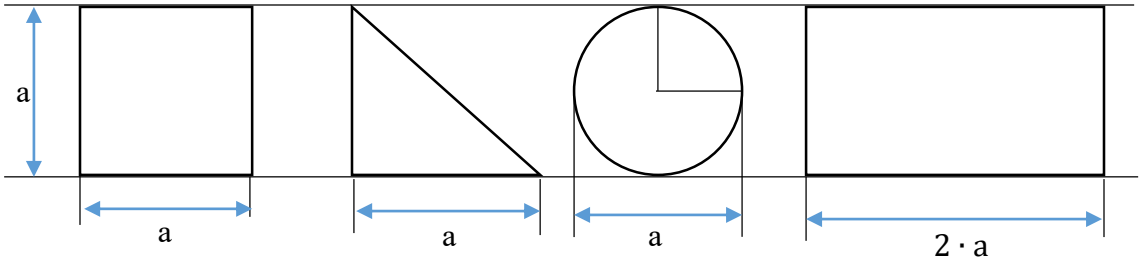


Fig. 25 Cross-sections.

Parameter a is set-up to 40 mm. As was derived before, the twist angle is

$$\theta_n = \frac{T \cdot L}{G \cdot J_{\rho,numeric}}.$$

where $J_{\rho,numeric}$ is obtained from the numerical solution. Verification of the numerical model can be restricted directly to this parameter. More details you can find in [6].

Its analytical equivalents are:

- for circular cross-section $J_\rho = \frac{\pi D^4}{32}$,
- for square cross-section $J_\rho = 2{,}25H^4$,
- for rectangular cross-section $J_\rho = a \cdot b\left[\frac{16}{3} - 3.36\frac{b}{a}\left(1 - \frac{b^4}{12 \cdot a^4}\right)\right]$,
- for triangular cross-section $J_\rho = 0.0915\, b^4\left(\frac{a}{b} - 0.8592\right)$.

More details you can find in [6].

The behavior of relative error

$$|J_\rho - J_{\rho,numeric}|/|J_\rho|$$

for variable number of elements is in the Tab. 4, and also it is visualized in Fig. 26.

Number of elements nx,ny	10	20	30	40	50
Square relative error %	3.204	0.840	0.393	0.236	0.163
Triangular relative error %	4.048	0.974	0.425	0.241	0.159
Circular relative error %	0.538	0.121	0.052	0.029	0.018
Rectangular relative error %	1.903	0.545	0.290	0.201	0.160

Tab. 4 Different type of cross-section with relative errors depending on number of elements.

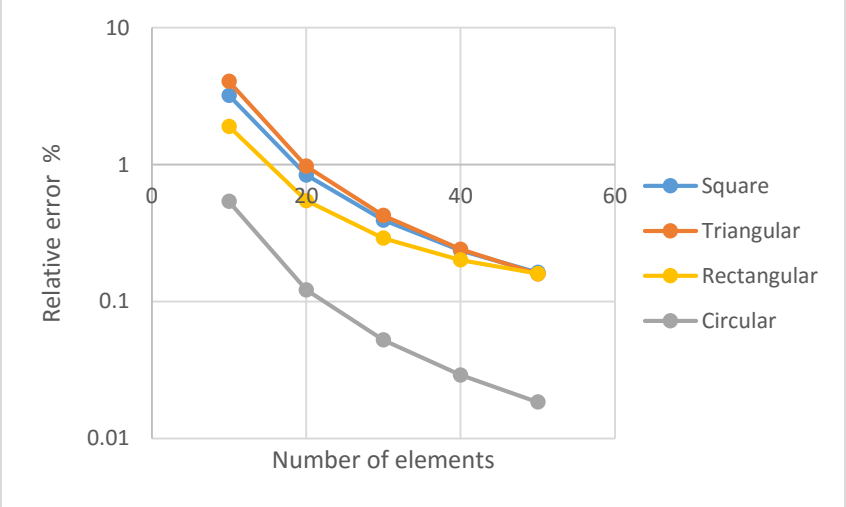


Fig. 26 Dependence of different number of elements with relative errors for different type of cross-sections.

The numerical results show, the application written in Python provides a sufficient approximation of the characteristics relating to torsion analysis. Then it can be used for the cases for which the analytical solution is not obtainable.

6. Solution of cases with unknown analytical solution

6.1. Description of the problem

The main goal of this chapter is to solve a case of torsion of prismatic bar with the cross-section without known analytical solution. After that the same example will be solved in the program Python and at last will be compared the results to see deference between of them. The chosen geometry of the cross-section is shown in the Fig. 27. Considered dimensions are height h, width b, length l_0 and radius R.

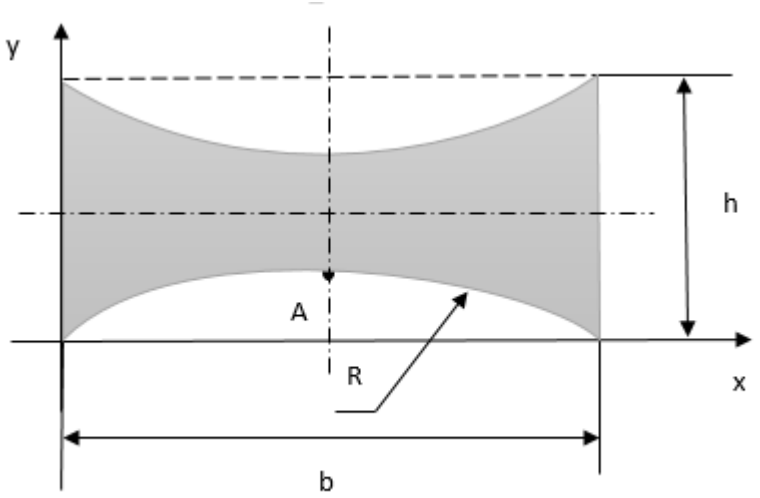


Fig. 27 Definition of the geometry of the cross-section.

There was used a linear elastic isotropic material model with assumption of these elastic constants: Young modulus E=200000MPa and Poisson's ratio μ=0.3.

6.2. Solution in ANSYS

The computational model is shown in the Fig. 28. MPC algorithm was used in order to apply appropriate boundary conditions in the FE model. Two pilot nodes were created. There was considered a unite torque (T=1Nm=1000Nmm) in the pilot node situated at the left hand side (coordinates are $[b/2, h/2, l_0 + 10]$). The model was fixed (zero values for all DOF) in all directions on the opposite end (coordinates are $[b/2, h/2, 10]$).

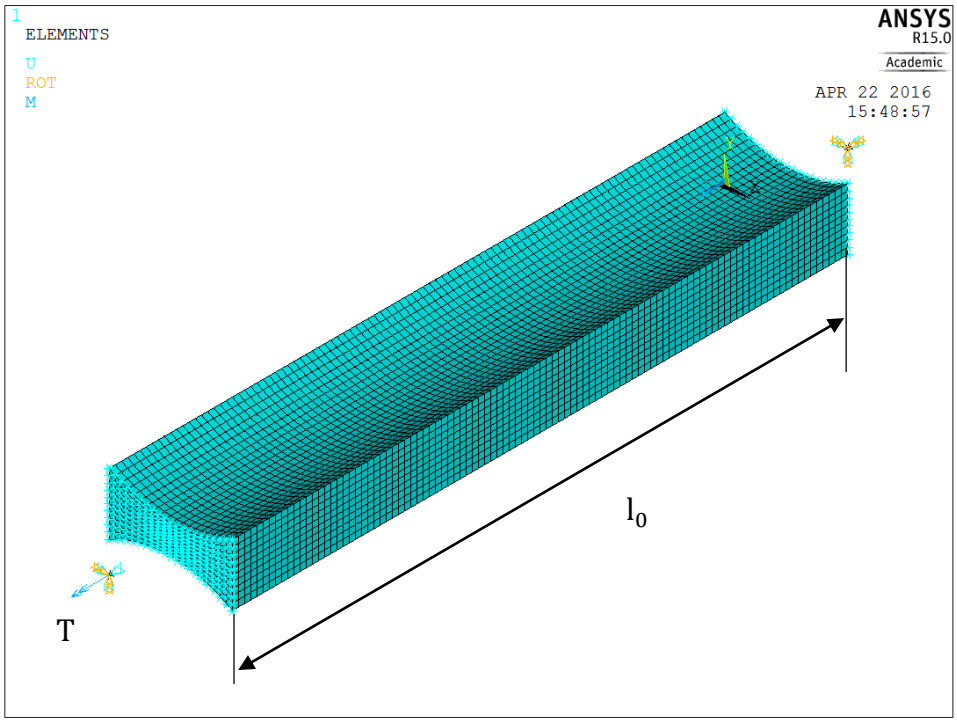


Fig. 28 FE mesh with prescribed boundary conditions.

A macro using APDL was coded in a text file, which enables to solve various cases of the geometry (different ratios h/b), see APPENDIX 1. The mesh shown in the Fig. 28 was created in such a way, that the default element size was assumed to be equal to h/10. The PLANE182 element type was considered to prepare the uniform mesh in a cross-section. Then the area was expanded along a line in order to get the whole 3D model (using SOLID185 elements). The evaluation of results was done in the section placed in the middle of the bar. The group of elements considered for evaluations was selected. It is shown in the Fig. 29. More details you can find in [7].

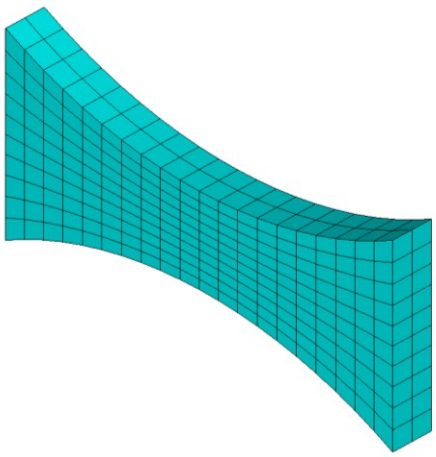


Fig. 29 Components of elements considered for evaluation of results in the middle of model.

Evaluation of results was performed also in ANSYS. The task was solved for 7 levels of h/b ratio (its name is k in the macro): 0.25; 0.27; 0.3; 0.4; 0.5; 0.75 and at last 0.95. The value of b was fixed in all calculations and was equal to 20mm. In the macro, R and h were considered as variables and we changed them in the Macro. The first solved case corresponds to the values: k=0.25, R=350mm, $h = k \cdot b$ and length $l_0 = 10 \cdot h$.

Now, the method of further evaluation should be described. When R was equal 350mm the hot-spot was not in the middle of the bar (Fig. 30), but the maximum value of the shear stress is placed in the middle for the case R=375mm, see Fig. 31.

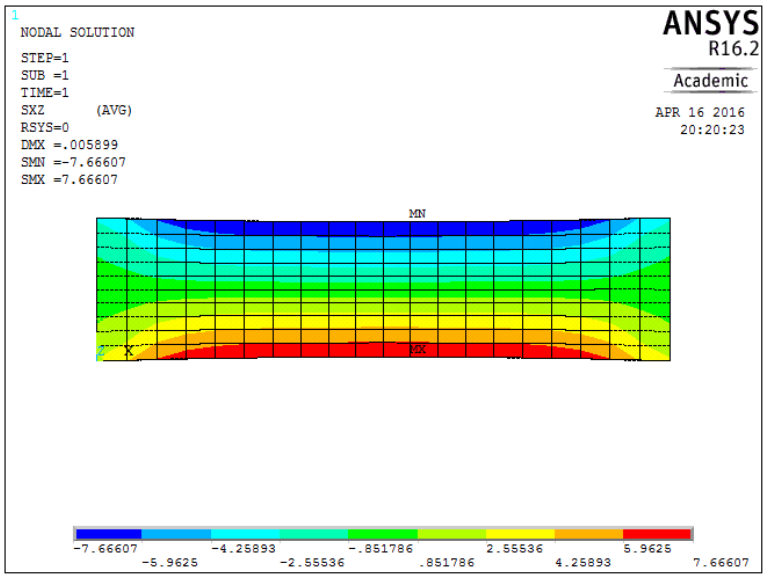


Fig. 30 Shear stress τ_{xy}[**MPa**] shown in contours for the case R=350 mm.

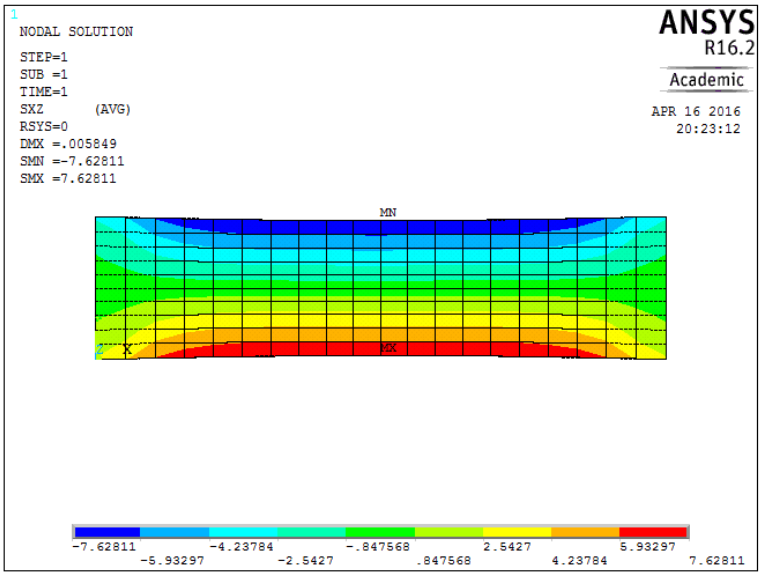


Fig. 31 Shear stress τ_{xy}[**MPa**] shown in contours for the case R=375 mm.

Thus there is always a value of ratio k corresponding to the change of critical stress value placement and the goal of our computations was to found this value for each level. The strategy was to increase the value of R by 1 step by step to do it. For instance, in the first case we obtained the value of R=370mm as can be seen in the Fig. 32.

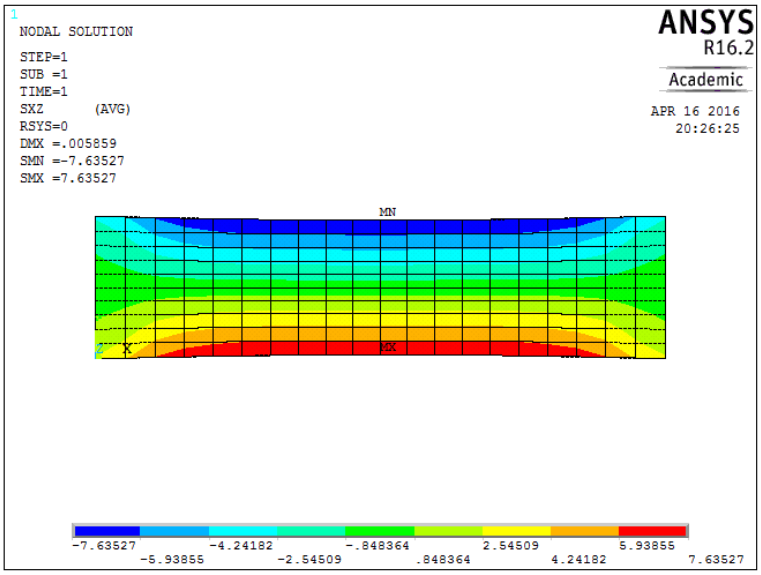


Fig. 32 Shear stress τ_{xy}[**MPa**] shown in contours for the case R=370 mm.

The same steps were followed for the case k=0.27. The limit values of the radius, which defines the searching interval were chosen as 240mm and 260mm. Finally the target value was founded as 250mm. Corresponding results are shown in the form of shear stress contours in the Fig. 33.

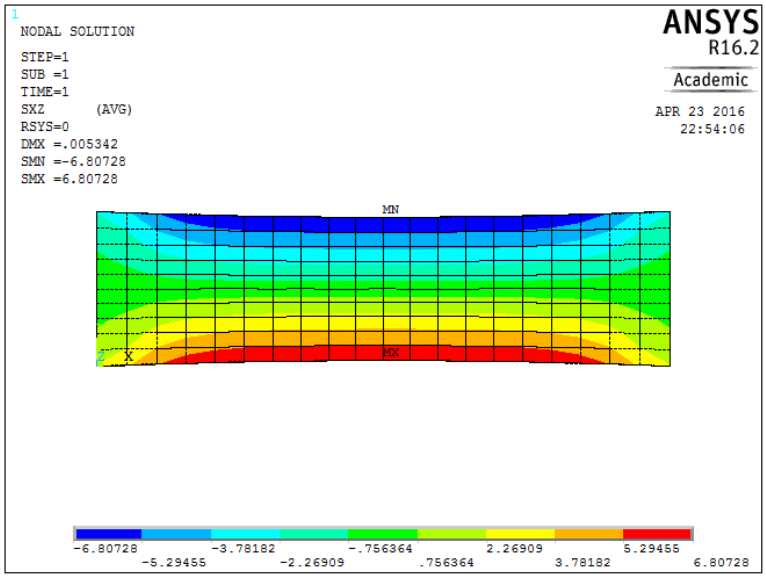


Fig. 33 Shear stress τ_{xy}[**MPa**] shown in contours for the case R=250 mm.

Third was for the case k=0.3. The limit values of the radius, which defines the searching interval were chosen as 150mm and 160mm. Finally the target value was founded as 155mm. Corresponding results are shown in the form of shear stress contours in the Fig. 34.

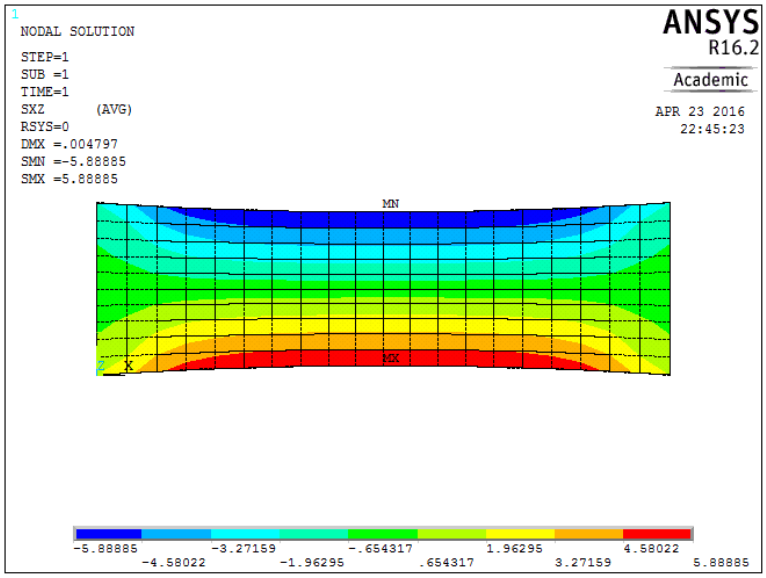


Fig. 34 Shear stress τ_{xy}[**MPa**] shown in contours for the case R=155 mm.

Fourth was for the case k=0.4. The limit values of the radius, which defines the searching interval were chosen as 50mm and 65mm. Finally the target value was founded as 55mm. Corresponding results are shown in the form of shear stress contours in the Fig. 35.

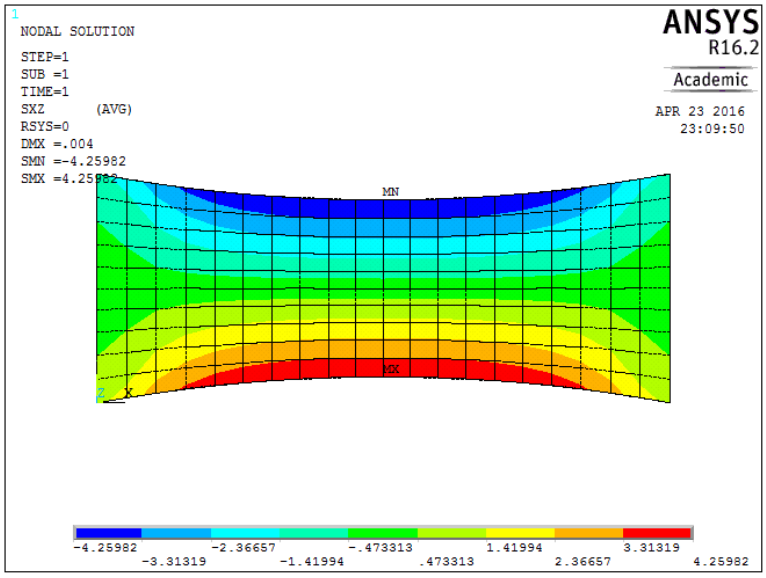


Fig. 35 Shear stress τ_{xy}[**MPa**] shown in contours for the case R=55 mm.

Fifth was for the case k=0.5. The limit values of the radius, which defines the searching interval were chosen as 25mm and 35mm. Finally the target value was founded as 30mm. Corresponding results are shown in the form of shear stress contours in the Fig. 36.

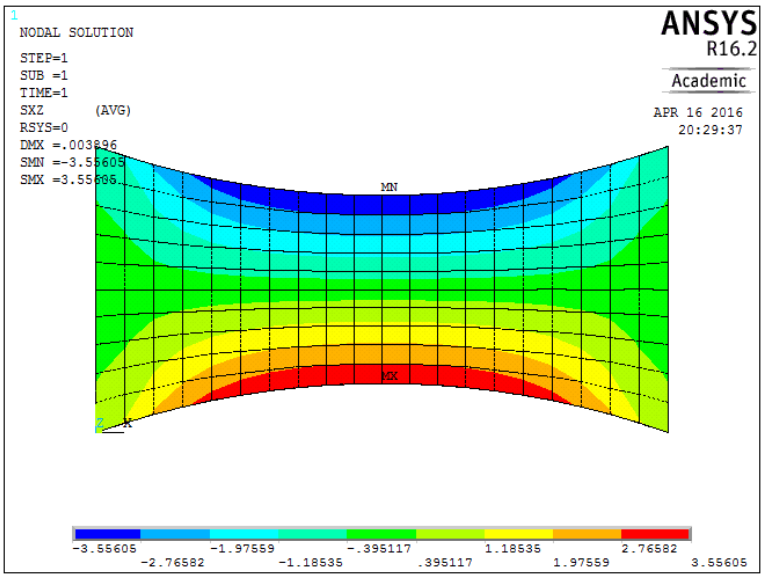


Fig. 36 Shear stress τ_{xy}[**MPa**] shown in contours for the case R=30 mm.

Next case was k=0.75 and same steps were followed. The limit values of the radius, which defines the searching interval were chosen as 12mm and 16mm. Finally the target value was founded as 14mm. Corresponding results are shown in the form of shear stress contours in the Fig. 37

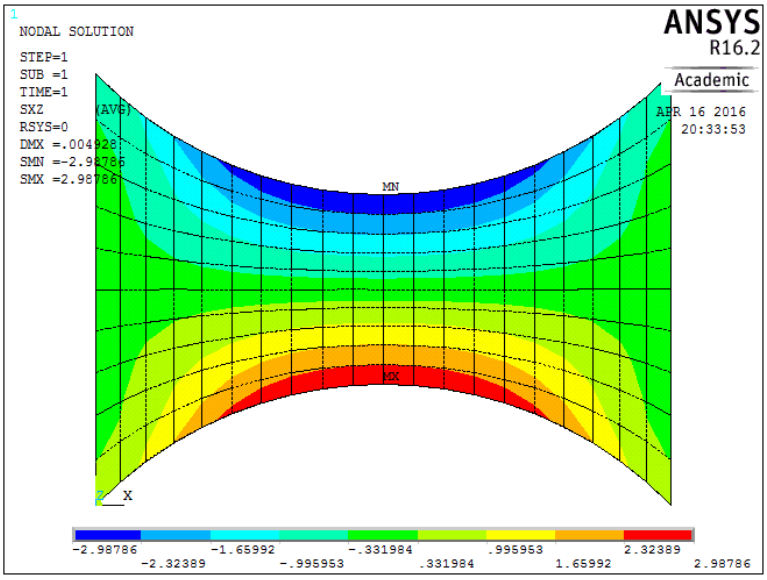


Fig. 37 Shear stress τ_{xy}[**MPa**] shown in contours for the case R=14 mm.

And last level k=0.95. The limit values of the radius, which defines the searching interval were chosen as 11mm and 13mm. Finally the target value was founded as 11.5mm. Corresponding results are shown in the form of shear stress contours in the Fig. 38.

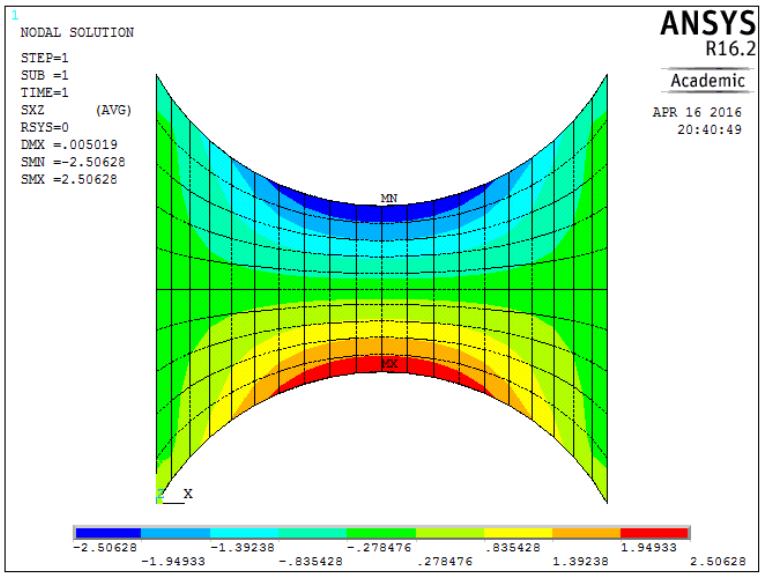


Fig. 38 Shear stress τ_{xy}[**MPa**] shown in contours for the case R=11.5 mm.

The deplanation of the cross-section for the case k=0.95 is shown on the Fig. 39. The displacement values are maximal in corners and their values are exactly the same in opposite corners.

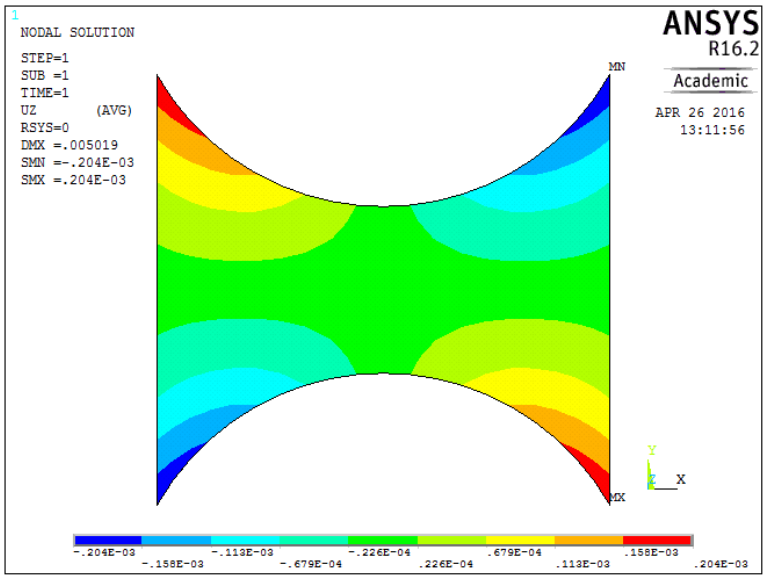


Fig. 39 Contours of displacement in Z direction [mm] for the case R=11.5 mm.

The numerical study leads to creation of resulting graph showing limit curve for the considered cross-section geometry. The graph is defined as dependence of R/b on the ratio h/b. The curve is obvious from the key diagram in the Fig. 40.

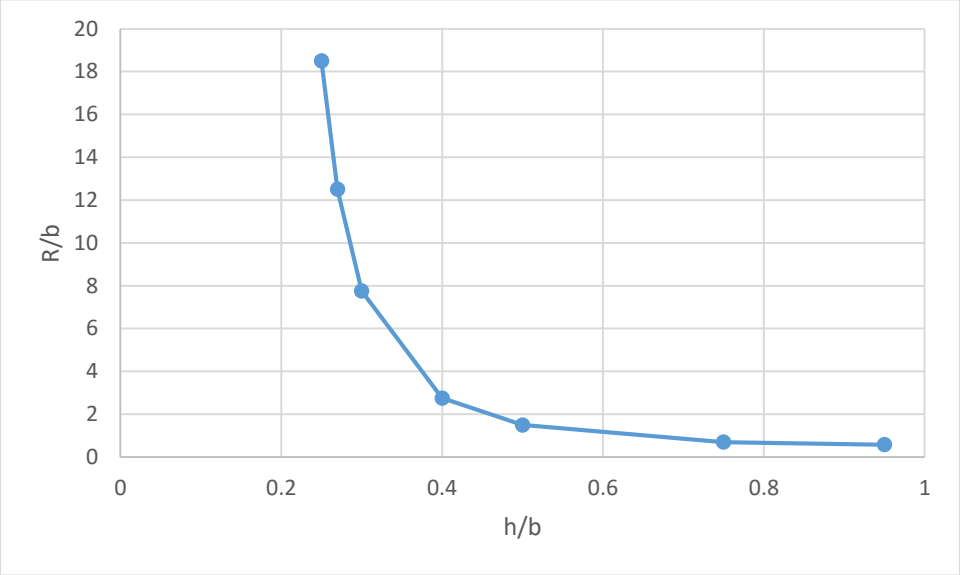


Fig. 40 Limit curve for considered geometry of cross-section.

ANSYS did not converge for the case k=1, because there is a zero thickness and in the narrowest part of the cross-section. The situation is clear from the Fig. 41where the geometry of the cross-section is considered for the case $R/b = 10/20 = 0.5$. A circumscribed area corresponds to a square.

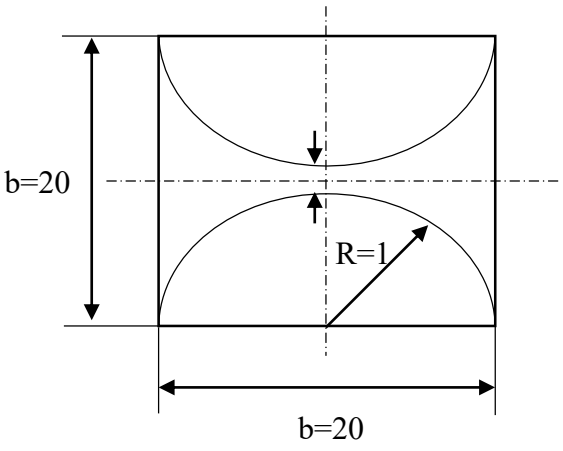


Fig. 41 Scheme of limit case for considered cross-section.

6.3. Solution using Python

The main goal of previous chapter was to solve a case of torsion of prismatic bar with the cross-section without known analytical solution – all as realistic 3 dimensional problem. Now the same case will be solved with program Python. The task will be to reconstruct the previous ANSYS solutions, again for 7 levels of h/b ratio: 0.25; 0.27; 0.3; 0.4; 0.5; 0.75 and at last 0.95. The value of b is fix in all calculations and is equal to 20mm. R and h are consider as variables and we change them in the program. The first solved case corresponds to the values: k=0.25, R=330mm, $h = k \cdot b$ and length $l_x = b;\ \ l_y = h; nx = 50$. The triangular elements were used.

Now, the method of further evaluation should be described. When R was equal 330mm the hot-spot was not in the middle of the bar (Fig. 42), but the maximum value of the shear stress is placed in the middle for the case R=340mm, see Fig. 43.

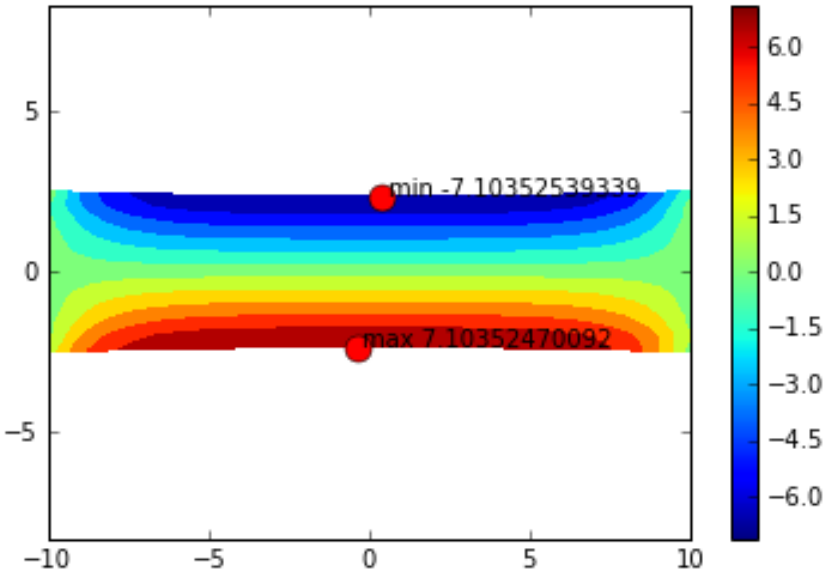


Fig. 42 Shear stress **τ[MPa]** shown in contours for the case R=330 mm.

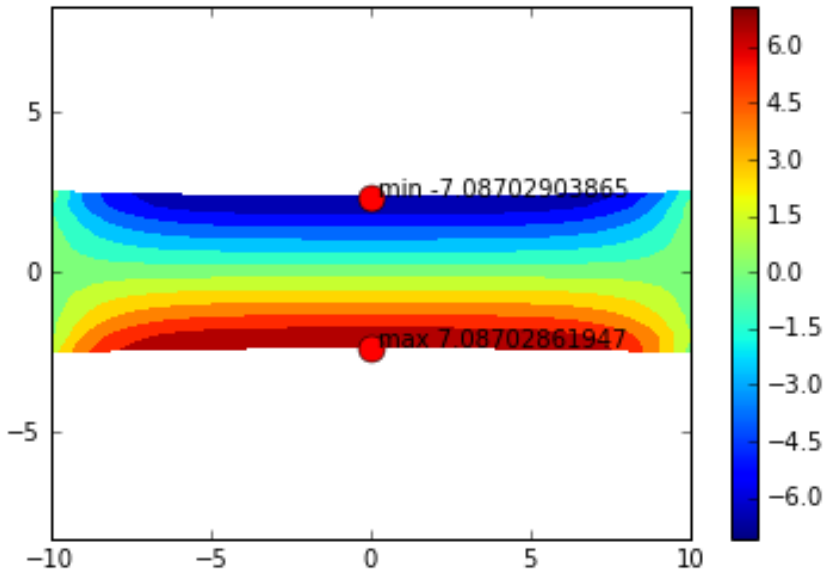


Fig. 43 Shear stress **τ[MPa]** shown in contours for the case R=340 mm.

Thus there is always a value of ratio k corresponding to the change of critical stress value placement and the goal of our computations was to found this value for each level. The strategy was to increase the value of R by 1 step by step to do it. For instance, in the first case we obtained the value of R=334mm as can be seen in the Fig. 44.

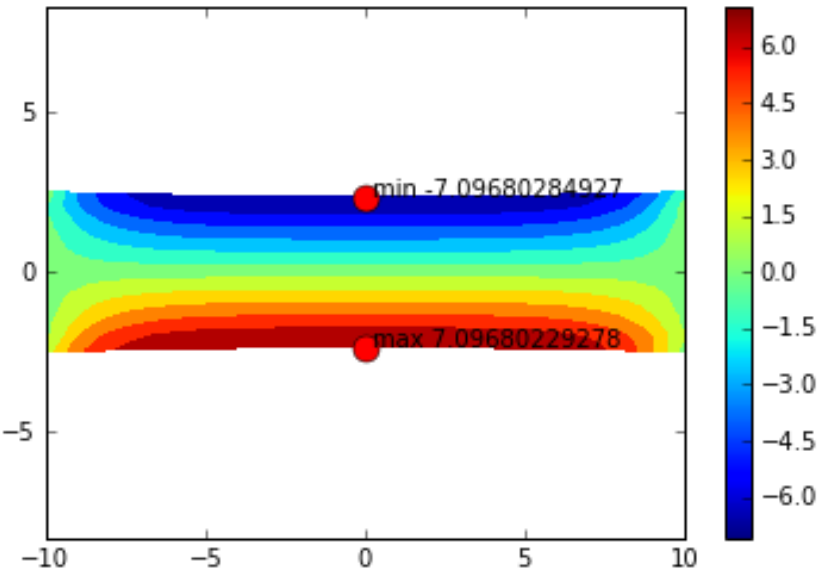


Fig. 44 Shear stress **τ[MPa]** shown in contours for the case R=334 mm.

The same steps were followed for the case k=0.27. The limit values of the radius, which defines the searching interval were chosen as 230mm and 240mm. Finally the target value was founded as 237mm. Corresponding results are shown in the form of shear stress contours in the Fig. 45.

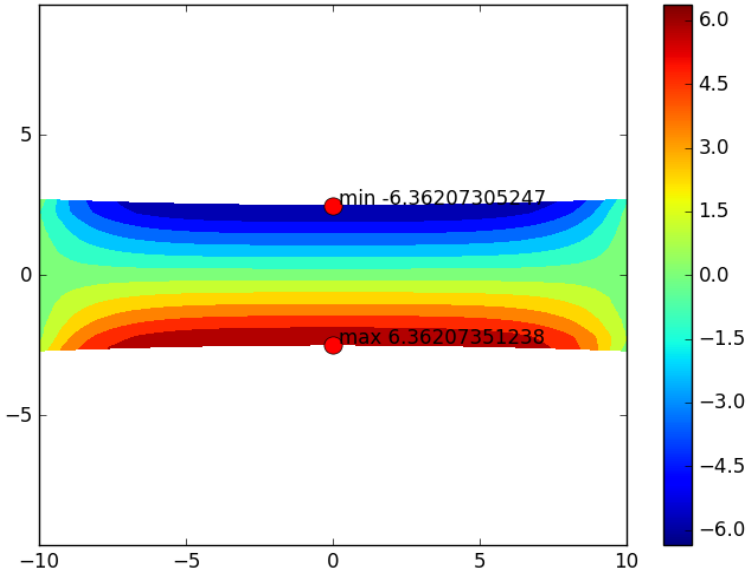


Fig. 45 Shear stress **τ[MPa]** shown in contours for the case R=237 mm.

Third was for the case k=0.3. The limit values of the radius, which defines the searching interval were chosen as 145mm and 150mm. Finally the target value was founded as 147mm. Corresponding results are shown in the form of shear stress contours in the Fig. 46.

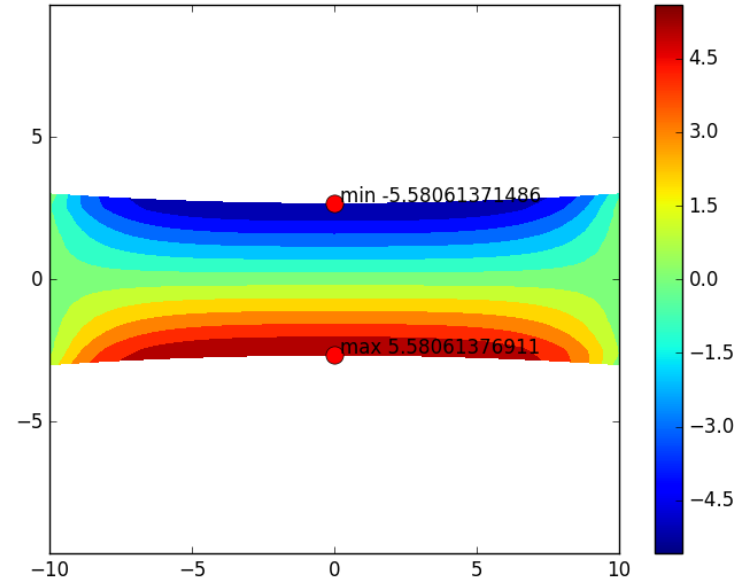


Fig. 46 Shear stress **τ[MPa]** shown in contours for the case R=147 mm.

Fourth was for the case k=0.4. The limit values of the radius, which defines the searching interval were chosen as 50mm and 55mm. Finally the target value was founded as 54mm. Corresponding results are shown in the form of shear stress contours in the Fig. 47.

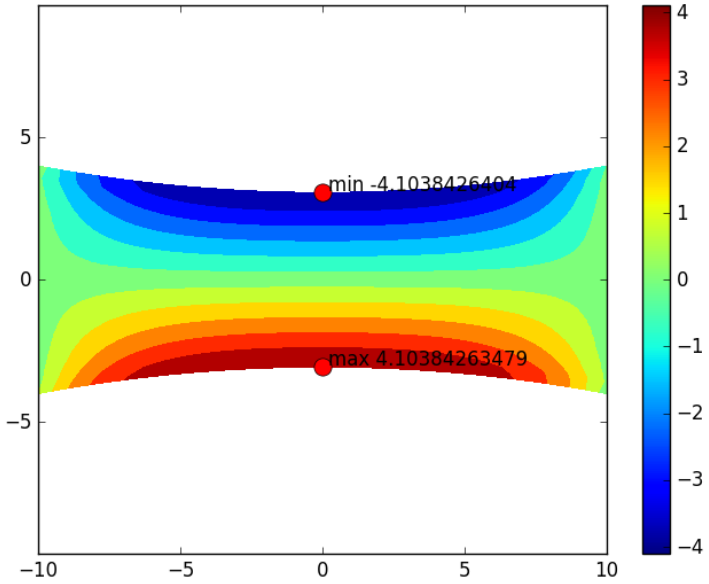


Fig. 47 Shear stress **τ[MPa]** shown in contours for the case R=54 mm.

Fifth was for the case k=0.5. The limit values of the radius, which defines the searching interval were chosen as 25mm and 30mm. Finally the target value was founded as 29mm. Corresponding results are shown in the form of shear stress contours in the Fig. 48.

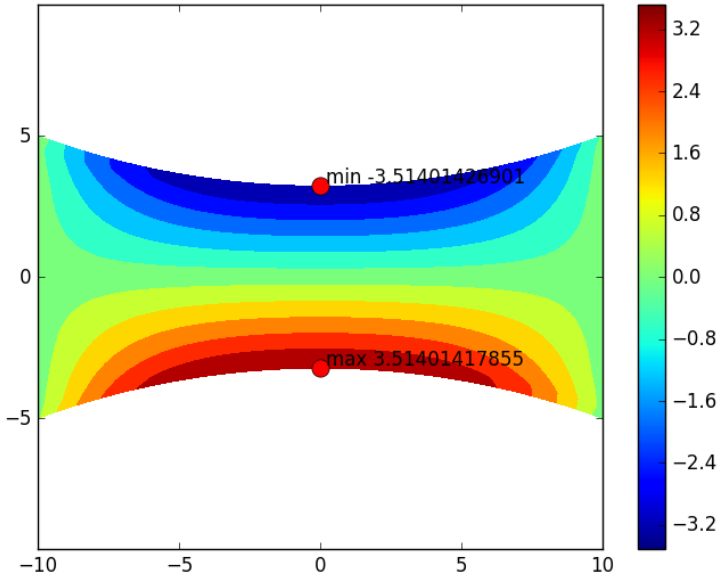


Fig. 48 Shear stress **τ[MPa]** shown in contours for the case R=29 mm.

Next case was k=0.75 and same steps were followed. The limit values of the radius, which defines the searching interval were chosen as 12mm and 16mm. Finally the target value was founded as 14mm. (Exactly same as in ANSYS) Corresponding results are shown in the form of shear stress contours in the Fig. 49.

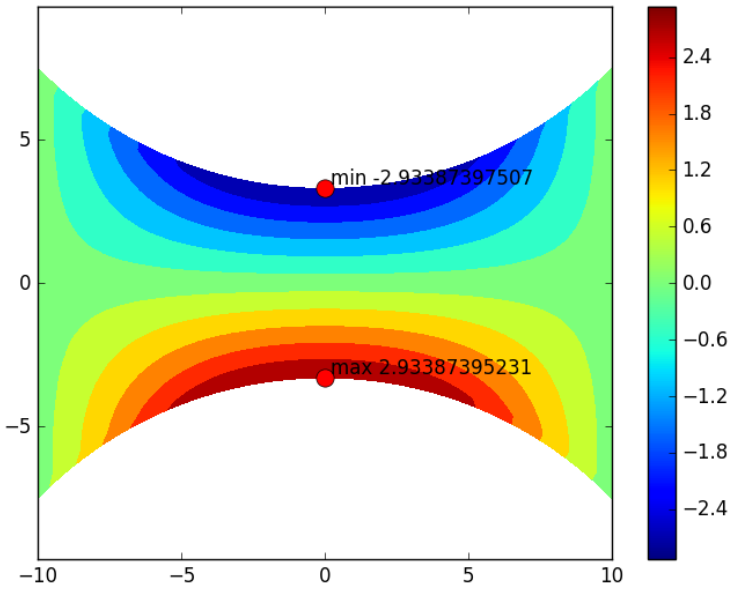


Fig. 49 Shear stress **τ[MPa]** shown in contours for the case R=14 mm.

And last level k=0.95. The limit values of the radius, which defines the searching interval were chosen as 11mm and 13mm. Finally the target value was founded as 11.5mm. (Exactly same as in ANSYS) Corresponding results are shown in the form of shear stress contours in the Fig. 50.

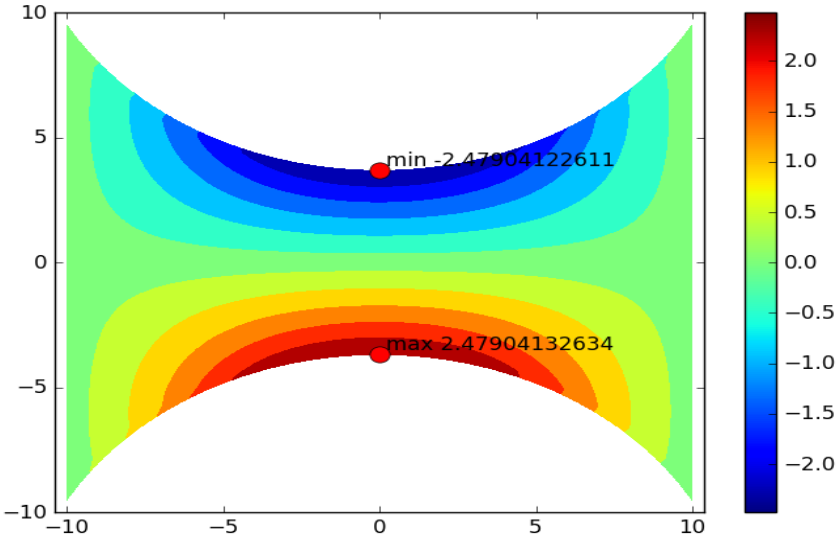


Fig. 50 Shear stress **τ[MPa]** shown in contours for the case R=11.5 mm.

The numerical study leads to creation of resulting graph showing limit curve for the considered cross-section geometry. The graph is defined as dependence of R/b on the ratio h/b. The curve is obvious from the key diagram in the Fig. 51.

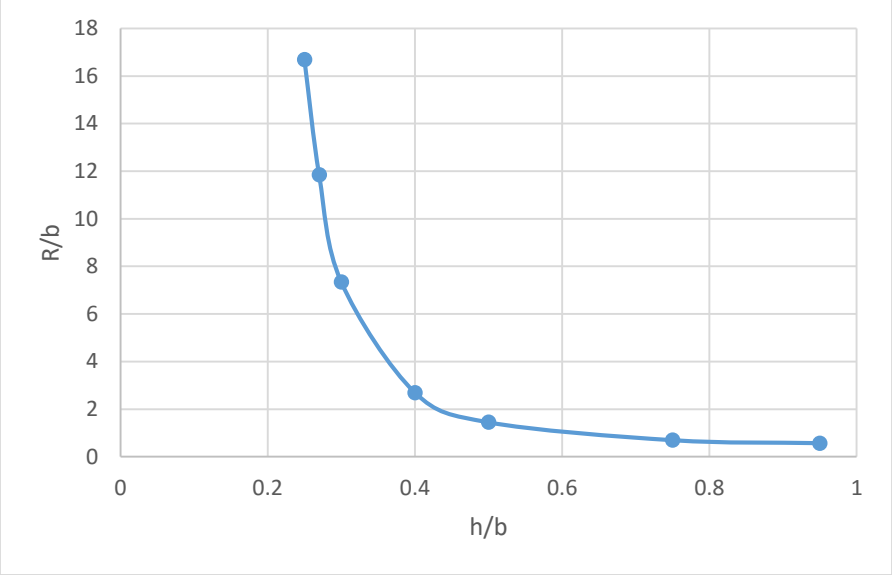


Fig. 51 Limit curve for considered geometry of cross-section.

For more distinguished we can see solutions of limit curves in ANSYS and in Python together see Fig. 52

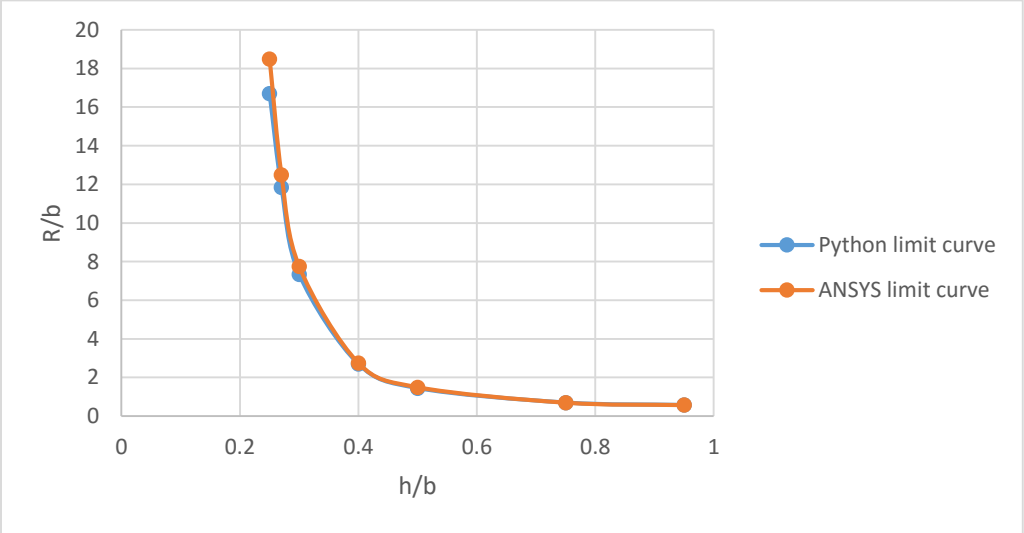


Fig. 52 Limit curves of ANSYS and Python approach

Before when we were solving relative errors in Python the number of nodes in x axis was 80 and in y axis 40, when we were solving same problem in ANSYS the number of nodes in x axis was 20 and in y axis 10. Results we can compare both of the in Fig 51. But we know that if we will have more nodes than before, example in x axis 100 and in y axis 50 the results will be closer real solutions. See Fig. 53.

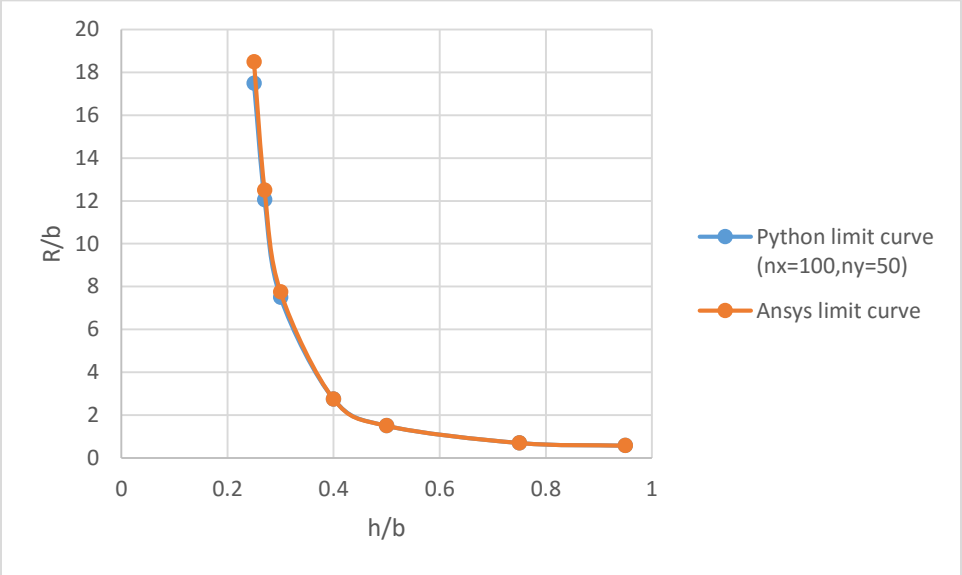


Fig. 53 Limit curves of ANSYS and Python approach, when we have increased nodes.

The results show, the Python application based on simplified model provides very similar data compared to the results obtained by ANSYS.

7. Conclusion

This master's thesis aims to solve the torsion of prismatic beams with non-circular cross-sections. At the beginning of this work, the theory of unconstrained torsion was introduced to get required partial differential equations in 2 dimensional space. Despite the simplification (reduction of the problem from 3 to 2 dimensional space), to get the analytical solution can be very difficult, even impossible. Therefore, the numerical method are employed. This thesis shows utilization of Finite Element Method (deformation variant). Beside the usage in own library written in Python, the thesis shows also the derivation of basic planar elements (triangular, and rectangular element).
The quality of the results for unconstrained torsion was verified for cases, the analytical solution is known (circle, square, rectangle, and triangle).
In the last part, the thesis deals with the realistic 3 dimensional problem with specific cross-section. The results of constrained torsion obtained from ANSYS are confronted with the data calculated in Python. The comparison shows, the Python library based on the reduced (2d) model (membrane analogy) can be taken as a good tool to analyze realistic twisted prismatic beam.

8. References

[1] "www.colorado.edu," University of Colorado, [Online]. Available: http://www.colorado.edu/engineering/CAS/courses.d/Structures.d/IAST.Lect07.d/IAST.Lect07.pdf.

[2] S. TIMOSHENKO and J.N. GOODIER, THEORY of ELASTICITY, NEW YORK TORONTO LONDON McGRAW-HILL BOOK COMPANY INC, 1951.

[3] P. N. P. J. Jan Francu, "TORSION OF A NON-CIRCULAR BAR," University of Technology, Brno, 2012.

[4] "www.wceng-fea.com," Valmont West Coast Engineering, 2008. [Online]. Available: http://www.wceng-fea.com/FEM.htm.

[5] "International Journal of Applied Mathematical Research," Science Publishing Corporation, 2015.

[6] "www.roymech.co.uk," 2010. [Online]. Available: http://www.roymech.co.uk/Useful_Tables/Torsion/Torsion.html.

[7] "ANSYS® Academic Research, Release 15.0, Help System, ANSYS documentation: Mechanical APDL, ANSYS, Inc.".

[8] HALAMA, Radim; FRYDRÝŠEK, Karel, "Using Strain and Force FEM Variants for the Saint-Vénanat Theory of Torsion," in *In Transactions of VŠB-TU Ostrava, mechanical series*, Ostrava, VŠB-TU Ostrava ISBN 80-7078-875-5, 2000, pp. 79-84.

[9] FRYDRÝŠEK, Karel; HALAMA, Radim, in *Using Strain & Force FEM Variants for the Saint Venant Theory of Torsion. In Proceedings of 9th ANSYS User's Meeting-Czech Republic and Slovakia - section A (Mechanical)*, Brno, Třešť, SVS FEM s.r.o, 2001, pp. 1-6.

9. Appendices

Appendices 1.

```
FINISH
/CLEAR,START
/PREP7
ET,1,PLANE182
!*
ET,2,SOLID185
!*
!*
MPTEMP,,,,,,,,
MPTEMP,1,0
MPDATA,EX,1,,200000
MPDATA,PRXY,1,,0.3
torque=1000
k=0.95
*SET,b,20
h=k*b
*SET,R,11.5
*SET,L0,10*h
*SET,yc,sqrt(R**2-(b/2)**2)
RECTNG,0,b,0,h,
CYL4,b/2,-yc,R
CYL4,b/2,h+yc,R
FLST,3,2,5,ORDE,2
FITEM,3,2
FITEM,3,-3
ASBA,       1,P51X
elsize=h/10
RATIOY=-0.9
LESIZE,2, , ,h/elsize,RATIOY, , , ,0
LESIZE,4, , ,h/elsize,RATIOY, , , ,0
LESIZE,13, , ,h/elsize,-RATIOY, , , ,0
LESIZE,14, , ,h/elsize,-RATIOY, , , ,0
LESIZE,15, , ,h/elsize,-RATIOY, , , ,0
LESIZE,16, , ,h/elsize,-RATIOY, , , ,0
LCCAT,13,14
LCCAT,15,16
AMESH,4
TYPE,   2
EXTOPT,ESIZE,L0/elsize,0,
EXTOPT,ACLEAR,1
```

```
!*
EXTOPT,ATTR,0,0,0
K, ,,,L0,
LSTR,1,5
CM,_Y,LINE
*SET,_Z1,LSINQR(0,13)
*IF,_z1,ne,0,then
LSEL,R,LCCA
*SET,_Z2,LSINQR(0,13)
LDELE,ALL
*SET,_Z3,_Z1-_Z2
*IF,_Z3,NE,0,THEN
CMSEL,S,_Y
CMDELE,_Y
*ENDIF
*ELSE
CMSEL,S,_Y
CMDELE,_Y
*ENDIF
VDRAG,      4, , , , , ,      5
/COM, CONTACT PAIR CREATION - START
CM,_NODECM,NODE
CM,_ELEMCM,ELEM
CM,_KPCM,KP
CM,_LINECM,LINE
CM,_AREACM,AREA
CM,_VOLUCM,VOLU
/GSAV,cwz,gsav,,temp
! Define surface-based constraint type of pair
MAT,1
R,3
REAL,3
ET,3,170
ET,4,175
KEYOPT,4,12,5
KEYOPT,4,4,1
KEYOPT,4,2,2
KEYOPT,3,2,0
KEYOPT,3,4,111111
TYPE,3
! Create a pilot node
N,23332, b/2,h/2,-10
TSHAP,PILO
```

```
E,23332
NSEL,S,,,23332
CM,pilotfix,NODE
CMSEL,S,_NODECM
! Generate the contact surface
ASEL,S,,,4
CM,_CONTACT,AREA
TYPE,4
NSLA,S,1
ESLN,S,0
NSLE,A,CT2 ! CZMESH patch (fsk qt-40109 8/2008)
ESURF
ALLSEL
ESEL,ALL
ESEL,S,TYPE,,3
ESEL,A,TYPE,,4
ESEL,R,REAL,,3
/PSYMB,ESYS,1
/PNUM,TYPE,1
/NUM,1
EPLOT
ESEL,ALL
ESEL,S,TYPE,,3
ESEL,A,TYPE,,4
ESEL,R,REAL,,3
CMSEL,A,_NODECM
CMDEL,_NODECM
CMSEL,A,_ELEMCM
CMDEL,_ELEMCM
CMSEL,S,_KPCM
CMDEL,_KPCM
CMSEL,S,_LINECM
CMDEL,_LINECM
CMSEL,S,_AREACM
CMDEL,_AREACM
CMSEL,S,_VOLUCM
CMDEL,_VOLUCM
/GRES,cwz,gsav
CMDEL,_TARGET
CMDEL,_CONTACT
/COM, CONTACT PAIR CREATION - END
!*
/COM, CONTACT PAIR CREATION - START
```

```
CM,_NODECM,NODE
CM,_ELEMCM,ELEM
CM,_KPCM,KP
CM,_LINECM,LINE
CM,_AREACM,AREA
CM,_VOLUCM,VOLU
/GSAV,cwz,gsav,,temp
! Define surface-based constraint type of pair
MAT,1
R,4
REAL,4
ET,5,170
ET,6,175
KEYOPT,6,12,5
KEYOPT,6,4,1
KEYOPT,6,2,2
KEYOPT,5,2,0
KEYOPT,5,4,111111
TYPE,5
! Create a pilot node
N,23333, b/2,h/2,l0+10
TSHAP,PILO
E,23333
NSEL,S,,,23333
CM,pilotload,NODE
CMSEL,S,_NODECM
! Generate the contact surface
ASEL,S,,,8
CM,_CONTACT,AREA
TYPE,6
NSLA,S,1
ESLN,S,0
NSLE,A,CT2 ! CZMESH patch (fsk qt-40109 8/2008)
ESURF
ALLSEL
ESEL,ALL
ESEL,S,TYPE,,5
ESEL,A,TYPE,,6
ESEL,R,REAL,,4
/PSYMB,ESYS,1
/PNUM,TYPE,1
/NUM,1
EPLOT
```

```
ESEL,ALL
ESEL,S,TYPE,,5
ESEL,A,TYPE,,6
ESEL,R,REAL,,4
CMSEL,A,_NODECM
CMDEL,_NODECM
CMSEL,A,_ELEMCM
CMDEL,_ELEMCM
CMSEL,S,_KPCM
CMDEL,_KPCM
CMSEL,S,_LINECM
CMDEL,_LINECM
CMSEL,S,_AREACM
CMDEL,_AREACM
CMSEL,S,_VOLUCM
CMDEL,_VOLUCM
/GRES,cwz,gsav
CMDEL,_TARGET
CMDEL,_CONTACT
/COM, CONTACT PAIR CREATION - END
/MREP,EPLOT
/SOLU
D,pilotfix, , , , , ,ALL, , , , ,
F,pilotload,MZ,torque
solve
NSEL,S,LOC,Z,L0/2-0.001,L0/2+0.001
CM,midnodes,NODE
ESLN,S
CM,midelems,ELEM
FINISH
/POST1
SET,LAST
PLNSOL, U,Z, 0,1.0
PLNSOL, S,XZ, 1,1.0
!these commands will work only for elsize=1
eplo
FLST,2,21,1
FITEM,2,512
FITEM,2,2888
FITEM,2,2987
FITEM,2,3086
FITEM,2,3185
FITEM,2,3284
```

```
FITEM,2,3383
FITEM,2,3482
FITEM,2,3581
FITEM,2,3680
FITEM,2,908
FITEM,2,4571
FITEM,2,4472
FITEM,2,4373
FITEM,2,4274
FITEM,2,4175
FITEM,2,4076
FITEM,2,3977
FITEM,2,3878
FITEM,2,3779
FITEM,2,611
!*
PATH,curvatur,21,30,20,
PPATH,P51X,1
PATH,STAT
!*
AVPRIN,0, ,
!*
PDEF, ,S,XZ,AVG
/PBC,PATH, ,0
!*
PLPAGM,SXZ,100,'NODE'
PLPATH,SXZ
/DSCALE,1,1.0
/REPLOT
PLNSOL, S,XZ, 1,1.0
```

Appendices 2.

```
#typeOfCS='square'   # 'circular'
#typeOfCS='circular'   # 'circular'
#typeOfCS='rectangular'  # 'rectangular'
#typeOfCS='triangular'    #  'triangular'
typeOfCS='rectangular_deformed'

# if plot_stress = False, the graph is the same as before
plot_stress = True         #  True or False
type_of_plotted_stress     = 'tau_xy' # 'norm_tau' 'tau_xz', 'tau_xy'
```

```
import numpy as np
import pylab as plt

if (typeOfCS=='square'):
   import rectangular_mesh as fem              # for square mesh CS

elif (typeOfCS=='rectangular'):
   import rectangular_mesh as fem            # for rectangle mesh CS

elif (typeOfCS=='triangular'):
   import circular_mesh as fem             # for triangle mesh CS
   flag_triag=True

elif (typeOfCS=='rectangular_deformed'):
   import rectangular_mesh as fem            # for rectangle mesh CS

elif (typeOfCS=='circular'):
   import circular_mesh as fem               # for cyrcle cross-sectin (CS)
   flag_triag=False

h = 1

PI = np.pi
volume_force_f = 1.0e0

def my_fun(x,y):
   return volume_force_f

if (typeOfCS=='square'):
   Lx = 40.0; Ly= 40.0;
   nx =50;
   ny = nx
   elements, coordinates, iGamma = fem.getMesh(Lx,Ly,nx,ny)
elif (typeOfCS=='rectangular'):
   Lx = 80.0; Ly= 40.0;
   ny = 50
   nx = ny*2;

   elements, coordinates, iGamma = fem.getMesh(Lx,Ly,nx,ny)

elif (typeOfCS=='triangular'):
   n = 50;R = 20.
   Lx = R
```

```
    Ly = R
    D = R*2
    elements, coordinates, iGamma = fem.getMesh(R,n,flag_triag)

elif (typeOfCS=='circular'):
    n = 50;R = 20.
    D = R*2
    elements, coordinates, iGamma = fem.getMesh(R,n,flag_triag)

elif (typeOfCS=='rectangular_deformed'):

    k = 0.25

    R = 350
    _b = 20
    _h = k* _b
    Lx = _b; Ly= _h;
    nx = 100

    ny = np.int(Ly/Lx*nx)

    elements, coordinates, iGamma = fem.getMesh(Lx,Ly,nx,ny)
    coordinates[:,0]-=0.5*Lx
    coordinates[:,1]-=0.5*Ly

    _H = 0.5*Ly + np.sqrt(R**2-(0.5*Lx)**2)

    print('_H',_H)

    _y = _H - np.sqrt(R**2-coordinates[:,0]**2)

    coordinates[:,1] =  2.0 * _y * coordinates[:,1]/Ly

    #typeOfCS='rectangular'
else:
    print('bad choice')

nEl = elements.shape[0]; n   = coordinates.shape[0]
F=np.zeros((n),dtype=np.float64)
K=np.zeros((n,n),dtype=np.float64)


# next block solves problem -(d^2u/dx^2 + d^u/dy^2) = 1
```

```
# to get function 'u(x,y)'

is_4node = typeOfCS=='rectangular' or typeOfCS=='square'
is_4node = False
Area_all = np.zeros(nEl)

if is_4node:
# !!! This part calculates Kh for 4 nodes rectangular elements
   GtG = np.zeros((4,4))
   M_int = np.zeros(4)
   for i in range(int(0.5*nEl)):
      ie0=elements[2*i,:]
      ie1=elements[2*i+1,:]
      ie = np.array([ie0[0],ie0[1],ie0[2],ie1[-1]])
      x=coordinates[ie,0]
      y=coordinates[ie,1]
      Ah=np.array([[1.0,x[0],y[0],x[0]*y[0]],\
                   [1.0,x[1],y[1],x[1]*y[1]],\
                   [1.0,x[2],y[2],x[2]*y[2]],\
                   [1.0,x[3],y[3],x[3]*y[3]] ])
      iAh=np.linalg.inv(Ah)
      _a = x[0]; _b = x[1]; _c = y[0]; _d = y[2]

      g11  = (_b-_a)*(_d-_c)
      g13  = 0.5*(_d**2-_c**2)*(_b-_a)
      g23  = 0.5*(_b**2-_a**2)*(_d-_c)
      g33  = (1.0/3.0)*((_b**3-_a**3)*(_d-_c)+(_d**3-_c**3)*(_b-_a))
      GtG[1,1] = GtG[2,2] = g11
      GtG[1,3] = GtG[3,1] = g13
      GtG[2,3] = GtG[3,2] = g23
      GtG[3,3] = g33

      Kloc=np.dot(iAh.transpose(),np.dot(GtG,iAh))
      K[np.ix_(ie,ie)]+=Kloc
      M_int[0] = g11
      M_int[1] = g23
      M_int[2] = g13
      M_int[3] = 0.25*(_b**2-_a**2)*(_d**2-_c**2)
      F_loc  = np.dot(iAh.transpose(),M_int)
      F[ie]+=F_loc
else:
# !!! This part calculates Kh for 3 nodes triangular elements
   dMhx=np.array([0.0,1.0,0.0]); dMhy=np.array([0.0,0.0,1.0])
```

```
    for i in range(nEl):
        ie=elements[i,:]
        x=coordinates[ie,0]
        y=coordinates[ie,1]
        Ah=np.array([[1.0,x[0],y[0]] , [1.0,x[1],y[1]] , [1.0,x[2],y[2]] ])
        iAh=np.linalg.inv(Ah)
        Ghx=np.array([np.dot(dMhx,iAh)]); Ghy=np.array([np.dot(dMhy,iAh)])
        Area_h=np.linalg.det(Ah)*0.5
        Area_all[i] = Area_h
        Kloc=np.dot(Ghx.transpose(),Ghx)+np.dot(Ghy.transpose(),Ghy)
        Kloc*=Area_h
        K[np.ix_(ie,ie)]+=Kloc
        xT = x.sum()/x.shape[0] # xT = (x0 + x1 + x2)/3
        yT = y.sum()/y.shape[0] # yT = (y0 + y1 + y2)/3
        Mh_xyT = np.array([1.0, xT, yT])
        F_loc   = np.dot(iAh.transpose(),Mh_xyT)*my_fun(xT,yT)*Area_h
        F[ie]+=F_loc

iInnerNew = np.arange(n)
iGammaNew = iGamma
iInnerNew = np.delete(iInnerNew, iGammaNew, None)
u=np.zeros(n)
u[iInnerNew]=np.linalg.solve(K[np.ix_(iInnerNew,iInnerNew)],F[iInnerNew])
u_max_abs=np.max(np.abs(u))

# second block provides integral which is the volume below the function 'u'
Volume = 0.0

du_dxy = np.zeros((nEl,2))

if is_4node:
# !!! This part calculates volume for 4 nodes rectangular elements
    for i in range(int(0.5*nEl)):
        ie0=elements[2*i,:]
        ie1=elements[2*i+1,:]
        ie = np.array([ie0[0],ie0[1],ie0[2],ie1[-1]])
        x=coordinates[ie,0]
        y=coordinates[ie,1]
        _a = x[0]; _b = x[1]; _c = y[0]; _d = y[2]
        Area_h=(_b-_a)*(_d-_c)
        uT = np.sum(u[ie])*(1./4)

        Ah=np.array([[1.0,x[0],y[0],x[0]*y[0]],\
```

```
            [1.0,x[1],y[1],x[1]*y[1]],\
            [1.0,x[2],y[2],x[2]*y[2]],\
            [1.0,x[3],y[3],x[3]*y[3]] ])
        iAh=np.linalg.inv(Ah)
        _a = x[0]; _b = x[1]; _c = y[0]; _d = y[2]

        g11  = (_b-_a)*(_d-_c)
        g13  = 0.5*(_d**2-_c**2)*(_b-_a)
        g23  = 0.5*(_b**2-_a**2)*(_d-_c)


        #Kloc=np.dot(iAh.transpose(),np.dot(GtG,iAh))
        #K[np.ix_(ie,ie)]+=Kloc
        M_int[0] = g11
        M_int[1] = g23
        M_int[2] = g13
        M_int[3] = 0.25*(_b**2-_a**2)*(_d**2-_c**2)

        Volume_i = np.dot(np.dot(iAh.T,M_int).T,u[ie])

        #print(Volume_i-Area_h*uT)
#       Volume+=Volume_i
        Volume+=Area_h*uT
#
else:
# !!! This part calculates volume for 3 nodes triangular elements
    for i in range(nEl):

        ie=elements[i,:]
        x=coordinates[ie,0]
        y=coordinates[ie,1]
        Ah=np.array([[1.0,x[0],y[0]] , [1.0,x[1],y[1]] , [1.0,x[2],y[2]] ])
        iAh=np.linalg.inv(Ah)
        Ghx=np.array([np.dot(dMhx,iAh)]); Ghy=np.array([np.dot(dMhy,iAh)])
        du_dx=np.dot(Ghx,u[ie])
        du_dy=np.dot(Ghy,u[ie])
        du_dxy[i,:] = [du_dx,du_dy]

        Area_h=np.linalg.det(Ah)*0.5
        uT = np.sum(u[ie])*(1./3)
        Volume+=Area_h*uT

        #xy_max
```

```
if (typeOfCS=='square'):
    print('...')
    Jrho_n= 1*Volume*4
    # if Lx = Ly = H
    a = Lx*0.5
    Jrho_a = 2.25*a**4

elif (typeOfCS=='rectangular'):
    a = Lx*0.5              #because a=Lx/2 (same was square CS) you wrote  a=Lx*0.5
    b = Ly*0.5
    Jrho_a=a*b**3*(16./3.-3.36*b/a*(1.-b**4./(12.*a**4)))  #semianalytical
    Jrho_n = Volume*4      #and here double integral u(x,y)dxdy*4
elif (typeOfCS=='triangular'):

    aa = np.sqrt(2.)*R
    bb = np.sqrt(R**2-(0.5*aa)**2)
    Jrho_n = 4*Volume
    #Jrho_a = (aa**3*bb**3)/(15.*aa**2+20*bb**2)
    Jrho_a = 0.0915*bb**4*(aa/bb-0.8592)
    #print(Jrho_a/Jrho_n)
elif ( typeOfCS == 'circular'):
    Jrho_a = np.pi*D**4/32
    Jrho_n = 4*Volume*4
    #print(Jrho_a/Jrho_n)
elif (typeOfCS == 'rectangular_deformed'):
    Jrho_n = Volume*4      #and here double integral u(x,y)dxdy*4
       print("Jrho_n:       ",Jrho_n)
if (typeOfCS != 'rectangular_deformed'):

    print("Jrho_a:       ",Jrho_a )
    relative_err = np.abs(Jrho_n-Jrho_a)/Jrho_a*100
    print("relative_error: ",relative_err,"%" )

################################  NOTES ######################################
#  the volume is only for one qurter, it has to be multiplied by factor 4
# Make the comparision for the circular shaft (R = 0.005 m)
# E = 2.1e11 Pa
# L = 0.5 m
# Mt = T = 20000 N*m
# mu = nu = 0.33
E = 2.0e5
```

```
L = 10 * h
Mt = 1e3
mu = 0.3
G = 0.5*E/(1.+mu)

 THETA = Mt/(G*Jrho_n)
gamma = THETA * L

const = 2*G*THETA

tau_plot = np.zeros(u.shape[0])
weigth   = np.zeros(u.shape[0])
for i in range(nEl):
   ie = elements[i,:]
   if type_of_plotted_stress=='norm_tau':
      norm_tau      = np.sqrt(du_dxy[i,0]**2+du_dxy[i,1]**2)*const
   elif type_of_plotted_stress=='tau_xz':
      norm_tau      = du_dxy[i,0]*const
   elif type_of_plotted_stress=='tau_xy':
      norm_tau      = du_dxy[i,1]*const

   tau_plot[ie] += norm_tau*Area_all[i]
   weigth[ie]   += Area_all[i]
tau_plot/=weigth

tau_n_max = tau_plot.max()
print('tau n:       ',tau_n_max)

if typeOfCS!='circular':
   tau_n_max = tau_plot.max()
   print('tau n:       ',tau_n_max)
else:
   tau_n_max = tau_plot.max()
   ttmp = np.abs(tau_plot).max()
   print('tau n:       ',ttmp)
   tau_a_max = Mt*R/Jrho_a
   print('tau a:       ', tau_a_max)
   relative_err = np.abs(tau_a_max-ttmp)/tau_a_max*100
   print("relative_error: ",relative_err,"%" )
if plot_stress:
   tmp_plot = tau_plot
else:
   tmp_plot = u
```

```
import numpy as np
from mpl_toolkits.mplot3d import Axes3D
import matplotlib.pyplot as plt
import matplotlib.tri as mtri
from matplotlib import cm


fig = plt.figure(0);plt.clf()
if 0:
   ax = fig.add_subplot(111, projection='3d')
   triang = mtri.Triangulation(coordinates[:,0],coordinates[:,1], elements)

   ax.plot_trisurf(triang, tmp_plot, lw=0.2,cmap=cm.jet)#, edgecolor="black",
color="grey",alpha=0.5)

else:
   levels=np.linspace(tmp_plot.min(),tmp_plot.max(),12);
   plt.tricontourf(coordinates[:,0], coordinates[:,1], \
      elements, tmp_plot,edgecolor='b',levels=levels)
   plt.axis('equal')

if plot_stress:
   inMax = np.argmax(tmp_plot)
   xy_max = coordinates[inMax,:]
   plt.plot(xy_max[0],xy_max[1],'ro',markersize=11)
   plt.text(xy_max[0]+0.01*Lx,xy_max[1]+0.01*Ly, 'max '+str(tmp_plot[inMax]) )
   inMin = np.argmin(tmp_plot)
   xy_min = coordinates[inMin,:]
   plt.plot(xy_min[0],xy_min[1],'ro',markersize=11)
   plt.text(xy_min[0]+0.01*Lx,xy_min[1]+0.01*Ly, 'min '+str(tmp_plot[inMin]) )

m = cm.ScalarMappable(cmap=cm.jet)
m.set_array(tmp_plot)
plt.colorbar(m)



#plt.xlim([-0.5*Lx,0.5*Lx])
```

Circular mesh

```
import numpy as np
import pylab as plt

def getMesh(radius,n,flag_triag):
#n = 15  number of layers
   flagUseTri=True
#    flag_triag=True
# ------------------------------------------------------------------------#
   x = np.linspace(0,1.,n)
   y = np.linspace(0,1.,n)
   coordinates = np.zeros((n*n,2),dtype=np.float64)
   elements = np.zeros(((n-1)*(n-1),4),dtype=np.int32)
   levelOfLayer = np.zeros((n*n),dtype=np.float64)
   cnt = 0
   for j in range(n):
      for i in range(n):
         coordinates[cnt,:] = np.array([x[i],y[j]])
         if (i>j):
            levelOfLayer[cnt]=i
         else:
            levelOfLayer[cnt]=j
         cnt+=1
   cnt = 0
   for j in range(n-1):
      for i in range(n-1):
         elements[cnt,:]=np.array([i,i+1,i+1+n,i+n])+n*j
         cnt+=1
   for i in range(coordinates.shape[0]):
      xy=coordinates[i,:]
      n_xy = np.linalg.norm(xy)
      if n_xy > 0:
         xy=coordinates[i,:]/n_xy*(levelOfLayer[i]+1)/n
         coordinates[i,:]=xy
         if (flag_triag):
            a=(levelOfLayer[i]+1)/n
            x=a*xy[0]/(np.sum(xy))
            y=a-x
            coordinates[i,0]=x;
            coordinates[i,1]=y;
   if flagUseTri:
      elements=np.concatenate((elements[:,0:3],elements[:,[0,2,3]]))
   for i in range(elements.shape[0]):
```

```
        ie = np.concatenate((elements[i,:],[elements[i,0]]))
        #plt.plot(coordinates[ie,0],coordinates[ie,1],'k.-')

    if flag_triag:
        iGamma = ie
#       plt.plot(coordinates[:,0],coordinates[:,1],'.')
        iGamma = np.zeros(4*(n-1))

        cnt = 0
        for i in range(coordinates.shape[0]):
            x = coordinates[i,0]
            y = coordinates[i,1]
            if abs(x)<1e-4 or abs(y)<1e-4 or abs(x + y - 1) < 1e-4:
                iGamma[cnt]=i
                print('i=',i)
                cnt+=1

    else:
        R_i = np.sqrt(coordinates[:,0]**2+coordinates[:,1]**2)
        iGamma = np.abs(R_i-1)<1e-4
        iGamma = plt.find(iGamma)

    coordinates*=radius
    print(levelOfLayer)

    #plt.axis('equal')
    #plt.axis('off')
    #plt.show()
    return elements, coordinates, iGamma
```

Rectangular mesh

```
import numpy as np

def getMesh(Lx,Ly,nx,ny):
   nEl=nx*ny*2
   nNod=(nx+1)*(ny+1)
   elements=np.zeros((nEl,3),dtype=np.int32)
   coordinates=np.zeros((nNod,2),dtype=np.float32)
   hx=Lx/nx
   hy=Ly/ny

   cnt=0
   for j in range(ny):
      for i in range(nx):
         elements[2*cnt,:]=np.array([i,i+1,(nx+1)+i+1])+j*(nx+1)
         elements[2*cnt+1,:]=np.array([i,(nx+1)+i+1,(nx+1)+i])+j*(nx+1)
         cnt+=1
   cnt=0
   for j in range(ny+1):
      for i in range(nx+1):
         coordinates[cnt,:]=np.array([i*hx,j*hy])
         cnt+=1

   edge0=np.array([np.arange(0,nx),np.arange(1,nx+1)]).transpose()
   edge1=np.array([np.arange(nx+1,(nx+1)*(ny+1)-(nx+1)+1,nx+1),\
            np.arange(2*(nx+1),(nx+1)*(ny+1)+1,nx+1)]).transpose()-1
   edge2=np.array([np.arange(nx+1,1,-1),np.arange(nx,0,-1)]).transpose()+(nx+1)*ny-1
   edge3=edge1-nx;edge3[:,[0,1]]=edge3[:,[1,0]]

   e0=np.unique(edge0)
   e1=np.unique(edge1)
   e2=np.unique(edge2)
   e3=np.unique(edge3)

   e01 = np.concatenate((e0,e1))
   e23 = np.concatenate((e2,e3))

   e0123 = np.concatenate((e01,e23))
   e = np.unique(e0123)

   return elements,coordinates,e  #,edge0,edge1,edge2,edge3

getMesh(2.0,1.8,3,3)
```